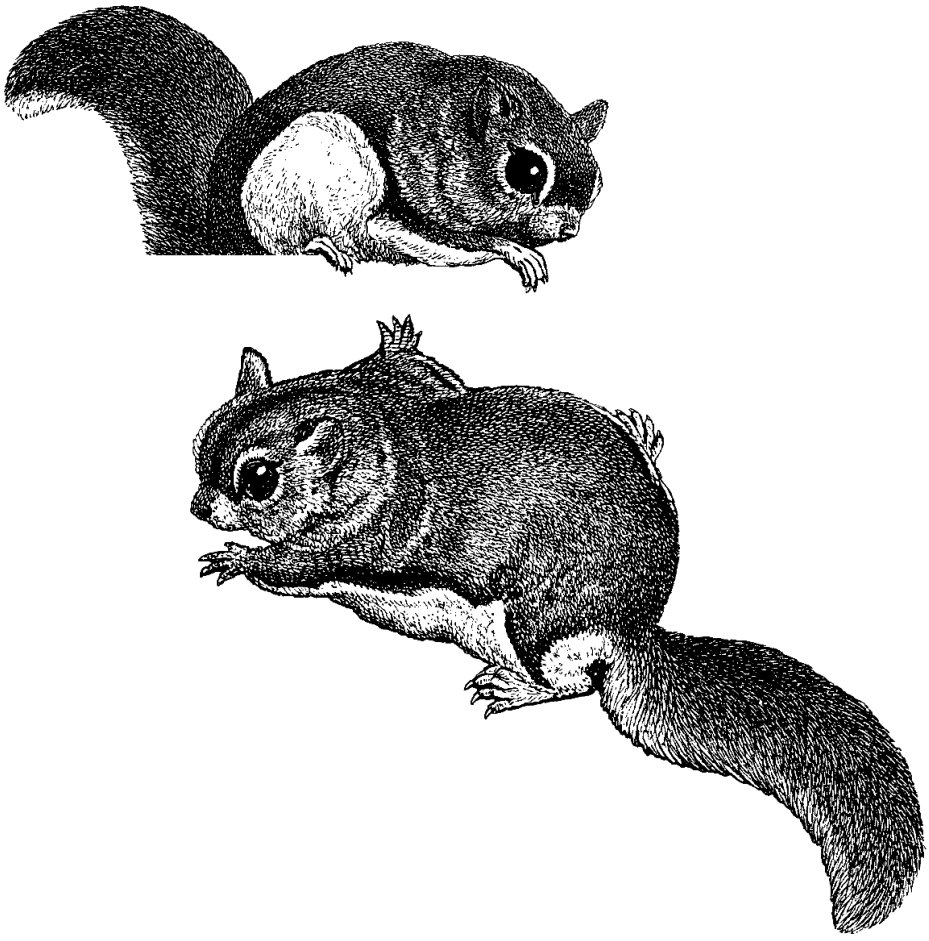
Robin Niksonun ***"Learning PHP, JavaScript and MySQL"*** kitabı əsasında hazırlanmışdır.



PHP

PHP

Mündəricat

DİQQƏT! BU VƏSAİT TAM HAZIR DEYİL!

1

Dinamik tərkibli veb-səhifələrə giriş

Ümumdünya hörümçək toru — daim inkişaf etdirilən fasiləsiz şəbəkədir. Bu şəbəkəni yaradalıması üçün keçən əsrin 90-cı illərində konsepsiya irəli sürülmüş və şəbəkənin yaradılması barədə yekun qərar verilmişdir. CERN yüksəktexnologiyalı sınaqlarda inanılmaz böyük həcmli məlumatlar əldə edirdi, hansı ki, dünyanın hər tərəfindən olan alimləri bu haqda məlumatlandırmaq lazım idi.

O vaxtı İnternet artıq mövcud idi və ona bir neçə yüz min kompüter qoşulmuşdu, buna görə Tim Berners-Li (CERN mütəxəssisi) hiperistinad mühitdən istifadə etməklə onların arasında naviqasiya üsulunu fikirləşdi. Hiperistinad (Hyper Text Transfer Protocol (HTTP)) ötürülmə protokoldur. O həmçinin hipermətn nişan dili (Hyper Text Markup Language (HTML)) adlandırılmış xüsusi

xüsusi dili yaratdı. Bütün bunlarla yanaşı, o tarixdə ilk brauzer və veb-serveri yaratmışdır hansı ki, bu xidmətlərdən hələdə istifadə olunur.

Amma o zaman bu konsepsiya inqilabçı xarakter daşıyırdı. Əvvəlcə əlaqələrin əsas həcmi elektron məlumat tablolarına zəng etmiş və qoşulmuş ev modemlərinin istifadəçilərinə aid idi və onlar ayrı-ayrı kompüterə əsaslanırdı. Bu xidmət vasitəsilə yalnız digər istifadəçilər ilə ünsiyyət saxlamaq və məlumat mübadiləsi etmək mümkün idi. Beləliklə, kolleqalarla və dostlarla effektiv elektron ünsiyyət qurmaq üçün çoxlu elektron məlumat tablolarının hər birinin istifadəçisi olmaq lazım idi.

Amma Berners-Li bir dəfəyə bütün bunları dəyişdirdi, və 1990-cı illərin ortasına artıq üç əsas rəqabət aparan bir-biri ilə qrafiklər 5 milyon ziyarətçi diqqətdən istifadə etmiş brauzer mövcud idilər. Amma tezliklə oldu görünür ki, koeçto qaçırdılmışdı. Əlbəttə, başqa səhifələrə keçid üçün hiperistinada malik olan mətn və qrafik səhifələr parlaq konsepsiya idi, amma nəticədə bu konsepsiya kontekstin dinamik dəyişikliyində istifadəçilərin vacib ehtiyaclarının təmin olunmasına görə kompüterlər və İnternetin cari potensialını əks etdirmirdi. Ümumdünya hörümçək toru olduqca ifadəsiz təəssürat bağışlayırdı. Hətta mətn effektləri və GIF-şəkil animasiyalarının tətbiqi heç nəyi dəyişdirmədi.

İstifadəçilərin, axtarış matorları və sosial şəbəkələrindən sıx istifadə etməsi nəticəsində Ümumdünya hörümçək toruna əhəmiyyətli dəyişiklər daxil edildi. Bu fəsildə vebin simasını formalaşdıran müxtəlif komponentlər, onların zənginləşdirilməsi və istifadəsinin yolları və həmçinin bu əməliyyatları yerinə yetirməyə imkan verən proqram təminatının qısa icmalı veriləcək.

Abreviaturalardan istifadə etmək vaxtı çatıb. Bunu etməzdən əvvəl, mən onların dəqiq izahatı verməyə çalışmışan. Amma abreviaturaların açılışını bilmirsinizsə, narahat olmağa dəyməz çünki, bütün

təfərrüatlar kitabın oxunması müddətində aydınlaşacaq.

HTTP və HTML: Berners-Li tərəfindən əsası qoyulmuş

HTTP sorğuların istiqamət sırasını və cavabların alınmasını nizama salan qarşılıqlı təsirin standartıdır.
Serverin işi ondan ibarətdir, müştəridən sorğunu qəbul edərək və ona məzmunlu cavabı versin. Adətən "cavab" dedikdə tələb olunmuş veb səhifə nəzərdə tutulur. Bu səbəbdən də "server" (xidmət edən) terminindən istifadə olunur. Serverlə müştəri qarşılıqlı təsirdə olur, elə buna görə də bu anlayış həm brauzerə, həm də brauzerin işlədiyi kompüterə aid edilir. Müştəri ilə server arasında bir sıra başqa qurğular yerləşə bilər, məsələn marşrutlayıcı, giriş modulları, şlüzlər və s. Bütün bunlar müştəri ilə serverin arasında sorğu və cavabların səhvsiz yerdəyişməsinin təminatı üzrə müxtəlif məsələləri yerinə yetirirlər. Bir qayda olaraq, bu informasiyanın göndərilməsi üçün İnternetdən istifadə olunur.
Adətən veb-server eyni anda bir neçə qoşulmanı emal edə bilər, lakin müştəri ilə əlaqə yoxluğunda veb server daxilolan qoşulmanın gözlənməsi rejimindədir. Qoşulma prosesində sorğunun daxil olması zamanı server göndərilən sorğunun alınmasını təsdiq edir.

"Sorğu — cavab" proseduru

Ən ümumi halda *"**sorğu** — **cavab**"* prosesi brauzer vasitəsilə veb-səhifənin tələb edilməsidir. Bundan sonra brauzer səhifənin vizual görünüşü ilə məşğul olur (şəkil. 1.1).
Bu halda bu əməliyyat ardıcıllığına riayət edilir.

1. Siz brauzerinin ünvan sətirinə http://server.com daxil edirsiniz.
2. Sizin brauzeriniz server.com adlı domenə uyğun olan IP-ünvan axtarır.
3. Brauzer server.com-un əsas səhifəsinə daxil olmaq üçün sorğu yollayır.

4. Sorğu İnternet üzərindən həyata keçirilir və server.com veb-serverinə inteqrasiya olunur.
5. Sorğunu almış veb-server öz sərt diskində tələb olunan veb səhifəni axtarır.
6. Server veb səhifəni tapır və onu əks marşrut üzrə brauzerin ünvanına göndərir.
7. Brauzer veb səhifəni əks etdirir.

Tipik veb səhifənin ötürülməsi zamanı bu proses serverdə olan hər bir obyekt üçün həyata keçirilir: videodan təşkil olunmuş qrafiki elementlər, Flash-çarx və hətta CSS şablonu.

Onu da bilin ki, 2-ci addımda brauzer server.com adlı domenə aid olan IP-ünvan axtarır. İnternetə hər qoşulduğunuzda, kompüteriniz müəyyən IP-si ünvanı var. Amma, bir qayda olaraq, veb-serverlərə giriş adlar üzrə həyata keçirilir, google.com kimi. Bilməlisiz ki, brauzer serverlə bağlı IP-ünvanı tapmaq üçün və sonra da bu IP-dən kompüterlə əlaqədə istifadə etmək üçün domen adlarının xidməti (Domain Name Service (DNS)) kimi adlandırılan köməkçi internet-xidmətinə müraciət edir.

Dinamik veb səhifələrin ötürülməsi zamanı böyük miqdarda əməliyyat prosedurları həyata keçirilir.
Dinamik ardıcıllıq üçün addımlar "sorğu —" belələr cavabdır.

1. Siz brauzerinin ünvan sətirinə http://server.com daxil edirsiniz.
2. Sizin brauzeriniz server.com adlı domenə uyğun olan IP-ünvanı axtarır.
3. Brauzer server.com-un əsas səhifəsinə sorğu göndərir.
4. Sorğu Şəbəkə üzərindən həyata keçirilir və server.com veb-serverinə inteqrasiya edilir.
5. Sorğunu almış veb-server öz sərt diskində veb səhifəni axtarır.
6. Əgər əsas səhifə yaddaşda yerləşdirilmişmə, veb-server ilk öncə səhifənin PHP-ssenariləri ehtiva edən faylla təqdim

edildiyini yoxlayır və belə olan halda səhifəni PHP şərhçi verir.

7. PHP şərhçisi PHP-kodu yerinə yetirir.
8. PHP kodunun bəzi fraqmentləri MySQL-təlimatlarını ehtiva edə bilər. Belə olan halda isə PHP şərhçisi, öz növbəsində, MySQL məlumat bazasının prosessoruna verir.
9. MySQL məlumat bazası təlimatın icrasının nəticələrini PHP şərhçisinə qaytarır.
10. PHP şərhçisi veb-serverə PHP kodunun icrasının nəticələrini qaytarır, həmçinin MySQL məlumat bazasından alınmış nəticələr.
11. Veb-server səhifəni sorğu göndərən müştəriyə qaytarır, hansı ki, ekranda bu səhifəni əks olunur.

Əlbəttə, bu proseslə tanış olmaq və üç elementin birgə işi haqqında bilmək yaxşıdı, lakin təcrübədə bu təfərrüatlar bizə lazım olmayacaq, çünki, hər şey avtomatik rejimdə həyata keçirilir.

PHP, MySQL, JavaScript və CSS-dən istifadənin üstünlükləri

Bu fəsilin başlanğıcında Web 1.0 texnologiyasının əhatəsi təqdim edilmişdi, lakin çox vaxt keçmədən Web 1.1 texnologiyasının yaradılması ilə bu sahədə sıçrayış əldə olundu. Belə ki, artıq Java, JavaScript, JScript (Microsoft korporasiyası tərəfindən istehsal edilən JavaScript-in bir başqa variantı) və ActiveX kimi dillərdə brauzer genişlənmələri hazırlanmağa başlanıldı. Server nöqteyi-nəzərindən irəliləyiş ümumi şlüz interfeysinin (Common Gateway Interface (CGI)) yaradılması olmuşdurdur. ssenarilərin belə dillərindən istifadələr, Perl (alternativ PHP dilinə) kimi, və server tərəfində ssenarilərin icraları (ifaları) — bir faylın tərkibinin dinamik yerləşdirməsi (və ya sistem çağırışı çıxış verilmişləri) başqa fayla.

Nə vaxt ki, vəziyyət qəti aydınlaşdı, qabaqcıl mövqelərdə üç əsas texnologiya qaldı. Baxmayaraq ki, Perl ssenarilərinin dili öz tərəfdarlarının hesabına məşhurluğu saxladı, lakin PHP dilinin sadəliyi və MySQL məlumat bazası proqramına inteqrasiya edilmiş

istinad mümkünlüyünə görə bu dilin istifadəçilərin sayı üzrə ikiqat artdı. Amma stillərin (Cascading Style Sheets (CSS)) və və HTML-ın kaskad cədvəllərinin dinamik manipulyasiyasında istifadə edilən tənliyin ən əhəmiyyətli tərkib hissəsi sayılan JavaScript hal-hazırda müştəri tərəfindən qoyulan ən mürəkkəb məsələlərinin həllinə yönəlmiş AJAX-proseslərini öz üzərinə götürür. Veb səhifənin AJAX texnologiyası sayəsində məlumatlar emal edilir və sorğular istifadəçini xəbərdar etmədən fon rejimində veb-serverə göndərilir.

Şübhəsizdir ki, PHP ilə MySQL-in xüsusi simbiozu onların inkişafına şərait yaradır, amma ilk növbədə istehsalçıları bu xidmətləri seçməkdə nə cəlb edir? Bunun sadə cavabı var: istifadyəsi yüngüldür - bu texnologiyalardan saytlarda dinamik elementlərin sürətlə yaradılması üçün istifadə etmək olar. MySQL sürətlidir və güclüdür, amma bu halda praktik olaraq bütün sayt üzrə axtarış və məlumatların işlənməsi üçün lazım olan məlumat bazasının sistem tərəfindən istifadəsində sadə təkliflər mövcuddur, hansı brauzerlər üçün nəzərdə tutulmuşdur. Məlumatların saxlanılma və bu məlumatların çıxardılması üçün PHP MySQL ilə birlikdə tətbiq edildikdə, siz əsas sosial şəbəkələrin saytlarının hazırlaması üçün təşkil edənləri, lazımları alırsınız və Web 2.0 texnologiyasına keçid üçün.

Və nə vaxt ki, siz həmçinin JavaScript-i və CSS-ı birlikdə birləşdirəcəksiniz, sizdə vısokodinamiçnıx və interaktiv saytlar yaradılması üçün resept çıxacaq.

PHP-dən istifadə

PHP-dən istifadə zamanı veb səhifələrin dinamik xüsusiyyətləri təmin edən vasitələrin yerləşdirilməsi əhəmiyyətli dərəcədə sadələşir. Səhifələrə PHP genişlənilməsi mənimsədildə, onlarda ssenari dilinə birbaşa yol meydana çıxır. Yalnız istehsalçıya buna oxşar kod yazmaq lazımdır:

```
<?php
echo "Today is ". date (" l"). ". ";
?>
Here's the latest news.
```

Açan teq <? php veb-serverə teqə qədər bütün sonrakı kodun interpretasiyasına icazəni verir? >. Hər şey, konstruksiya xaricində,

sadə HTML şəklində müştəri çatdırılır. Buna görə də brauzerdə sadəcə `Here's the latest news` mətni göstərilir. PHP-teqlərinin daxilində inteqrasiya edilmiş date funksiyası serverin sistem vaxtına uyğun olan həftənin cari gününü əks etdirir.
Nəticədə bu iki hissənin çıxışı təxminən aşağıdakı şəkildə alınır:

```
Today is Wednesday. Here's the latest news.
```

PHP — kifayət qədər elastik dildir və bəzi istehsalçılar bilavasitə PHP-konstruksiyanı PHP kodunun yanında yerləşdirməyə üstünlük verirlər:

```
Today is <? php echo date (" l");? >. Here's the latest news.
```

Həmçinin formatlaşdırma və informasiyanın nəticəsinin başqa üsulları mövcuddur. Onu bilmək lazımdır ki, PHP-dən istifadə edərək, veb-istehsalçılar ssenarilərin dilini alırlar, hansı ki, hərçənd və C-da kompilyasiya olunulmuş kodun tezliyinə malik deyil və ya ona belə dillər, amma yenə də işləyir inanılmaz sürətli və bundan başqa HTML nişanına çox yaxşı yazılır.

Əgər siz mənim nəzəriyyəm ilə paralel işləmək üçün bu kitabda göstərilən PHP nümunələrini yığmağa hazırlaşırsınızsa, kodun şərhçi tərəfindən emalını təmin etmək üçün kodu <? php və ? > teqlərinin daxilində yazmağı unutmayın. Bu məsələnin sadələşdirilməsi üçün qabaqcadan bu teqləri ehtiva edən example.php faylını hazırlamaq olar.

PHP-dən istifadə etməklə, siz veb-serverdə qeyri-məhdud idarəetmə imkanlarına malik olursunuz. Əgər HTML-də cəld dəyişikliklər etmək, kredit kartının məlumatlarını emal etmək, məlumat bazasına istifadəçi haqqında məlumatları əlavə etmək və ya kənar saytdan informasiya almaq lazımdırsa, bütün bunları özü HTML koddan təşkil olunmuş PHP-fayllarının köməyilə etmək mümkündür.

MySQL-dən istifadə

Əlbəttə, istifadəçilərin sizin saytınızla iş zamanı etdiyi dəyişikliklərin izlənilməsi, tam ölçüdə HTML çıxış kodunun dinamik dəyişikliyinin imkanları haqqında danışmaq olmaz. Əvvəllər Ümumdünya hörümçək torunda olan saytların çoxu belə məlumatların saxlanılması üçün strukturlaşdırılmayan mətn fayllarından istifadə edirdi. Amma belə yanaşma bir sıra problemlərə gətirib çıxarırdı, əgər fayl etibarlı ona eyni zamanda olan giriş vaxtı bir çox istifadəçi yaranan zədələrdən bloklanmamışdı. Bundan başqa strukturlaşdırılmayan fayl inanılmaz ölçülərə qədər genişlənə bilirdi ki, bu halda təbii ki, onlarla işləmək çətin idi.

Məhz bu cür hadisələrdən sığortalanmaq üçün sorğuların strukturlaşmış sistemi vasitəsilən nisbi məlumat bazalarından istifadə etmək tələbi meydana çıxır. Və MySQL, artıq böyük miqdarda veb-serverə quraşdırılmış pulsuz, mükəmməl məlumat bazası sistemidir. MySQL etibarlı, xüsusi sürətli idarə sisteminə malik və sorğularda sadə ingilis sözlərinə oxşar komandalardan istifadə edən məlumat bazalarından ibarətdir.

Məlumat bazası sizin məlumatlarınızı ehtiva edən bir və ya bir neçə cədvələ malik olan MySQL strukturunun yüksək səviyyəsidir. Fərz edək, siz users (istifadəçilər) adlı cədvəl üzərində işləyirsiniz, hansı ki, bu cədvəldə soyadlar — surname, adlar — firstname və elektron poçt ünvanları — email üçün qrafalar yaradılmışdır və indi daha bir istifadəçini əlavə etmək lazımdır. Bu əməlliyyatları etmək üçün aşağıdakı tətbiqi komandalardan istifadə etmək lazımdır:

```
INSERT INTO users VALUES ('Smith', 'John', 'jsmith@mysite.com');
```

Əlbəttə, daha əvvəl qeyd edildiyi kimi məlumat bazasının və cədvəlin yaradılması və həmçinin bütün lazımlı sahələrin quraşdırılması üçün başqa komandalardan da istifadə olunacaq, amma burada istifadə edilən INSERT komandası yeni informasiyanın məlumat bazasına əlavə edilməsinin sadəliyini göstərir. INSERT komandası 1970-ci illərin əvvəllərində hazırlanmış (Structured Query Language (SQL)) və xatirimdə qalan ən köhnə proqramlaşdırma dillərindən biri olan COBOL-un (sorğuların strukturlaşmış dili) nümunəsidir. Bununla belə bu üsul məlumat bazasına sorğular üçün effektivdir və uzun müddətdir ki, istifadə edilir. Həmçinin sadəcə yerinə yetirilir və məlumatların axtarışı.

Fərz edək ki, istifadəçinin elektron poçt ünvanı var və bu poçtun sahibinin adını tapmaq lazımdır. Bunun üçün aşağıdakı MySQL-sorğusunu daxil etmək olar:

```
SELECT surname, firstname FROM users WHERE email='jsmith@mysite.com';
```

Bundan sonra MySQL Smith-i, John-u və istənilən başqa cütlük adı qaytaracaq, hansı ki, məlumat bazasında elektron poçt ünvanı ilə bağlıdır.

Asanlıqla demək olar ki, MySQL-in INSERT və SELECT komandalarının imkanları sadə olmasına baxmayaraq geniş əhatə dairəsinə malikdir. Məsələn, bir çox müxtəlif meyara uyğun olaraq bir neçə cədvəl birləşdirmək, nəticələri tələb etmək, çoxluqdan onların verilməsinin sırasını (nizamını) seçib bişirantlar, qismən təsadüfləri tapmaq, əgər yalnız axtarılan (tələb olunan) sətirin hissəsi məlumdursa (məşhurdursa), nəticələrin konkret verilmiş miqdarını qaytarmaq və bir çox şeylər etmək.

PHP-dən istifadə zamanı bütün bu çağırışları MySQL proqramına və ya komanda sətirinin MySQL interfeysini icra edilməsinə ehtiyac duymadan bilavasitə MySQL-ə yönəltmək olar. Bu isə o deməkdir ki, məlumatların sizə lazım olan elementinə qədər tapmaq üçün, siz onların emalı və bir çox axtarış əməliyyatının həyata keçirilməsi üçün nəticələri massivlərdə saxlaya bilərsiniz. hansılardan ki, hər biri əvvəlki əməliyyatlarla qaytarılmış nəticələrdən asılıdır.

Sonra görəcəksiniz ki, vermə üçün hələ (daha) düz MySQL-a bir neçə (bir qədər) əlavə funksiya ən böyük gücə tikilmişdir, hansılar ki, ən tez-tez rast gəlinən (görüşən) əməliyyatlar və məlumatların işlənməsinin sürətinin artımı üçün doğurmaq (çağırmaq) olar.

JavaScript-dən istifadə

Bu kitabda baxılan üç əsas texnologiyadan ən köhnəsi olan ***JavaScript***— HTML-sənədin bütün elementlərinə ssenarilərdən girişin alınması üçün yaradılmışdı. Başqa sözlə, bu istifadəçi ilə dinamik qarşılıqlı təsir vasitələri ilə təmin edir, məsələn məlumatların daxil edilməsi formalarında elektron poçt ünvanlarının məqbulluğunun yoxlaması, köməkçi və ya oxşar sətirlər (məs. "Siz həqiqətən bunu nəzərdə tuturdunuz?") və s. üçün (hərçənd təhlükəsizlik nöqteyi-nəzərindən, hansı ki, veb-serverdə həyata keçmək, bu texnologiyaya etibar etmək həmişə məcburdur olmaz).

CSS ilə JavaScript-i uyğunlaşdırılma dinamik veb səhifəsinin gücünün əsasını təşkil edir. Bununla belə brauzerlərin JavaScript-i icra etməsi müxtəlifliyi dilin reallaşdırılmasında əhəmiyyətli fərqlərlə gətirib çıxarır.

Əsasən bu fərqlər bəzi istehsalçıların öz rəqiblərinin məhsulları ilə uyğunluğuna diqqət vermədikdə və öz brauzerlərinə əlavə funksional imkanlarını verməyə çalışdıqda yaranır.

Xoşbəxtlikdən, əksər istehsalçılar artıq öz məhsullarının tam uyğunluğun ehtiyacını dərk ediblər ki, bunun nəticəsində də artıq brauzerin spesifikasiyasına uyğun kod yazmağa ehtiyac yoxdur. Amma brauzerlərin milyonlarla köhnəlmiş nüsxəsi hələ də qalır və istifadə olunur və gələcəkdə də istifadə olunacaq. Bununla belə bu tip uyğunsuzluq problemlərinin həlli mövcuddur.

İndi biz bütün brauzerlər tərəfindən qəbul edilən adi JavaScript-koduna baxacağıq:

```
<script type=" text/javascript" >
document.write (" Today is " + Date());
</script>
```

Kodun bu fraqmenti brauzerə script teqinin daxilində olan hər şeyi JavaScript kodu icra etməyi əmr edir, sonra isə brauzer cari sənədə "Today is" mətnini yazıb, həmçinin JavaScript-in Date funksiyası istifadəçinin ƏS-indən alınmış tarixi göstərəcək. Təxminən aşağıdakı nəticə alınacaq:

```
Today is Sun Jan 01 2017 01:23:45
```

Əgər JavaScript-in konkret versiyasını göstərmək tələb olunmursa, onda, bir qayda olaraq, script teqinin daxilində "type= text/javascript" yazmaq olar.

Daha əvvəl qeyd edildiyi ki, əvvəlcə JavaScript HTML sənədin daxilində olan müxtəlif elementlərin dinamik idarə edilməsinə yönəldilmişdi. Amma bir çox hallarda JavaScript, AJAX texnologiyasının reallaşdırılması üçün tətbiq edilir. Bu anlayışdan fon rejimində veb-serverə daxiletmə proseslərinin işarəsi üçün istifadə

olunur. (Əvvəlcə bu termin "qeyri-sinxron JavaScript və XML" kimi — Asynchronous JavaScript and XML adlandırılırdı, amma indi bu adlandırma bir qədər köhnəlmişdir.)

AJAX — Web 2.0 texnologiyalarına uyğun olaraq (bu termin Timlə O'Reylini, bani tərəfindən və nəşriyyatın icraçı direktoru tərəfindən populyarlaşdırılmışdır əsasını təşkil edən, hansında ki, bu kitab ingilis dilində çıxdı), veb səhifədən istifadə zamanı avtonom proqramlarına bənzəyir başladılar, bir halda ki, onlar artıq tamamilə yükləmək lazım deyillər. Bunun yerinə AJAX-ın sürətli çağırışınlarında veb səhifənin ayrı-ayrı elementləri cəlb edilmiş ola bilər, məsələn, sosial şəbəkə saytında sizin şəkiliniz dəyişdirilmiş və ya piksel ilə əvəz edilmiş ola bilər, suala (məsələyə) cavab verərək hansında ki, klikləmək lazımdır. Tamamilə bu mövzu 18-ci fəsildə baxılacaq.

CSS-dən istifadə

CSS-nin üçüncü standartının yaranmasından sonra CSS dinamik interaktivlik səviyyəsi təklif edir, hansı ki, bu əvvəllər yalnız JavaScript-in köməyi ilə dəstəklənirdi. Məsələn, siz yalnız HTML-in ixtiyari elementinə həm stil verə, həmçinin onun ölçülərini dəyişdirmək, rənglər, sərhədlər, intervallar, həm də, bir neçə CSS sətirindən istifadə etməklə, öz veb səhifənizə animatik keçidlər və dəyişikliklər əlavə edə bilərsiniz.

CSS-ni çox sadə üsul ilə veb səhifənin başlanğıcında <style> və </style> teqlərinin arasında HTML səhifəyə tətbiq etmək olar:

```
<style>
p {
text-align:justify;
font-family:Helvetica;
}
</style>
```

Bu qaydalar teq mətnin ilk<p> dü dəyişdirəcəklər ki, onda olan abzaslar tamamilə hamarlanmışdı və onlar üçün Helvetica şriftini istifadə olunsun.

İndi isə HTML5

Indiyə qədər əvvəlki veb standartlarına müxtəlif istehsalçılar tərəfindən bir çox əlavələr edilmişdir. Məsələn, brauzerdə qrafiklər ilə işləmək üçün əlavə modullara müraciəti tələb etməyən sadə üsul - Flash-dan istifadə olunurdu. Həmçinin Flash vasitəsilə veb səhifələrdə audio və video panelləri qurulurdu. Bundan başqa, Flash-in inkişaf prosesində HTML-a gizlicə daxil olmuş bir çox qeyri-məqbul uyğunsuzluqları qeyd etmək olar.

Beləliklə, bütün bu nahamarlıqları düzəltməyə yönəlmiş, həmçinin inkişafın növbəti mərhələsində İnterneti Web 2.0 texnologiyasının hüdudlarına daşımaq və sadalanan çatışmazlıqları aradan qaldırılmaq üçün HTML-in yeni standartı yaradılmışdı. Bu standart HTML5 adlandırılmışdır və 2004-cü ildə Mozilla Foundation və Opera Software (iki məşhur brauzer istehsalçısı) standartın ilk layihəsini təşkil etdiyi andan istifadə edilməyə başlanılmışdır. Amma standartın son layihəsi 2013-cü ilin başlanğıcında veb-standartlara rəhbərlik edən World Wide Web Consortium (W3C) beynəlxalq təşkilatı tərəfindən təqdim edilmişdi.

Nəzərə alaraq ki, hazırlama prosesi doqquz ilə başa gəldi, düşünmək olar ki, burada son spesifikasiya haqqında söhbət gedir.

Artıq 2015-ci ildən HTML5.1 üzərində işlər gedir. Beləliklə, standartın versiyalarının sonsuz silsiləsi davam edəcək. HTML5.1-də dilə bir çox rahat təkmilləşdirmənin (əsasən bu qalın kətanlara aiddir) əlavə edilməsi planlaşdırılmasına baxmayaraq, bu versiyanın əsası yeni standart sayılan HTML5-dir. Elə bu standartı öyrənmək və üzərində işləməyi bacarmaq lazımdır. Bir halda ki, bu standartdan gələcək illərdə də istifadə ediləcək.

Bu standart həqiqətən də HTML-ə bir çox yeni (əvvəlki xüsusiyyətləri dəyişdirilmiş və ya silinmiş olduqca rahat arxitektura) daxil etmişdir. Qısası, bu standartın meydana çıxması ilə siz aşağıdakı dəyişikliklərin şahidi olursuz:

Yeni teqlər.

Bu standarta <nav> və <footer> kimi yeni elementlər daxil edilmişdir;

Yeni API.

Məsələn, qrafiki qalın kətanlarda yazı və rəsm çəkmək üçün istifadə edilən <canvas> elementi (qalın kətan), <audio> və <video> elementi, avtonom veb-proqramları, mikroməlumat və lokal anbar əlavə etmə qabiliyyəti əlavə edilmişdir;

Proqramlar.

İki yeni təsvir texnologiyası daxil edilmişdir: riyazi düsturların nümayişi üçün MathML (Math Markup Language — riyazi işarələmə dili)) və yeni element xaricində <canvas> qrafik elementlərinin yaradılması üçün SVG (Scalable Vector Graphics — miqyası böyüdülə bilən vektor qrafiki) əlavəsindən istifadənin mümkünlüyü. Amma MathML və SVG ixtisaslaşmış xarakteri daşıyır və özündə kifayət qədər xüsusiyyət ehtiva edir ki, onların təsviri üçün ayrıca kitab lazım olacaq, buna görə bu kitabda onlara ötəri baxılacaq. Bütün bunlar (və yalnız) 22-ci fəsildə baxılmayacaq.

Apache veb-serveri

Dinamik veb-texnologiyasında PHP, MySQL, JavaScript, CSS və HTML5-ə əlavə olaraq altıncı qəhrəman — veb-serverdir. Bizim kitabımızda güman edilir ki, bu veb-server Apache olacaq. Biz artıq, HTTP protokolu üzrə müştəri ilə server arasında informasiya mübadiləsi prosesində veb-serverin iştirak etdiyi barədə cüzi məlumatımız var və əslində bu prosesdə daha məxfi və genişmiqyaslı əməliyyatlar yerinə yetirir.

Məsələn, Apache yalnız HTML-fayllara xidmət etmir — o həmçinin geniş fayl spektri ilə - təsvir və Flash-çarx fayllarından tutmuş, MP3 audiofaylları, RSS-axınları (Really Simple Syndication — abunə üzrə sadə paylanma) və s. kimi fayllar ilə işləyə bilir. HTML-səhifədə veb-müştəri tərəfindən tapılmış hər bir element serverdə istənilir, hansı ki, sonra sorğu nəticəsində xidmət həyata keçirilir.

Amma bu obyektlər statik fayl olmamalıdır. GIF-formatın təsvirləri kimi. Bu fayllar proqramlarla (məs. PHP ssenarilər) yaradıla bilər. PHP sizin üçün təsvir və başqa faylları yarada bilər. PHP-nin bu xüsusiyyəti istər real vaxt rejimində, istərsədə sonrakı xidmət müqabilində yaradıla bilər.

Bunun üçün adətən Apache və ya PHP-da qabaqcadan kompilyasiya olunulmuş və ya proqramın icrası zamanı çağrılan modullar mövcüddur. Belə modullardan biri PHP-də yazılmış qrafik elementlərin yaradılması və emalı üçün istifadə edilən GD (Graphics Draw — qrafiki rəsm çəkilməsi) kitabxanasıdır.

Apache həmçinin geniş şəxsi modul arsenalını dəstəkləyir. Sizin üçün ən əhəmiyyətli PHP modullarına əlavə olaraq, təhlükəsizliyin təmin olunması ilə məşğul olan modullar olacaq. Başqa nümunələr kimi veb-serverə genişi emal etməyə icazə verən Rewrite modulu xidmət edə bilərlər URL-ünvanların tiplərinin diapazonu və onları onun daxili tələblərinə uyğun olaraq yenidən yazmaq, və Proxy modulu, hansı ki, gizli yerdən tez-tez sorğu elənən səhifələrin xidməti üçün istifadə etmək olar, serverə yüklənməni azaltmaq üçün.

Kitabın sonrakı bölmələrində üç əsas texnologiya ilə verilən xüsusiyyətlərin yaxşılaşması üçün bu modulların praktiki tətbiq etməsi göstəriləcək.

Açıq mənbə kodlu proqramlar haqqında bir neçə söz

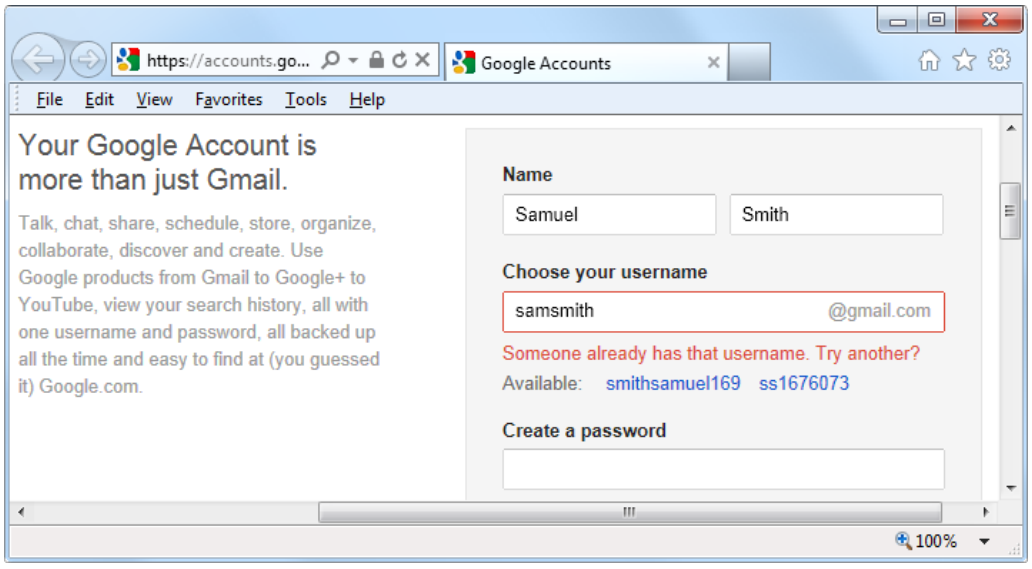
Aktiv mübahisə mövzusudur ki, bu texnologiyaların məşhurluğu onların açıq mənbə kodlu proqramlar ilə təqdim edilməsidir. Dəqiq deyə bilmərik bunu səbəbi nədir, amma həqiqətən də PHP, MySQL və Apache öz sinifində ən tələb edilən alətdir. Açıq kodlu proqramlar barədə onu deyə bilərik ki, bu tip proqramlar proqramçı komanda cəmiyyətində hazırlanmışdır və uyğun olaraq bu proqramın kodları açıq olduğuna görə istənilən şəxs koda öz istəklərini və ehtiyaclarını müvafiq olaraq xüsusiyyətləri verə bilər. Qısası bu tip proqramların mənbə kodları ümumi baxış və dəyişiklik üçün əlçatandır.

Bu proqramların bir üstünlüyü də bu kateqoriyada olan bütün proqramlar pulsuz istifadə oluna bilər. Əgər siz öz saytının keçirtmə qabiliyyətini yetişdirirsinizsə və onun xidmətinə müxtəlif serverləri cəlb edirsinizsə, əlavə lisenziyaların əldə edilməsi haqqında fikirləşmək lazım deyil. Həmçinin öz büdcəsinə yenidən baxmaq lazım deyil sistemin yenilənməsi və bu məhsulların ən son versiyalarının quraşdırılması haqqında qərar qəbul etməzdən əvvəl.

Bunlar birlikdə...

PHP, MySQL, JavaScript, CSS və HTML5-in həqiqi gözəlliyi bir çox məqamda gözə çarpır. Məsələn onlar birlikdə dinamik veb-

məzmunun istehsalı üzərində işləyirlər: veb-serverdə əsas iş ilə PHP məşğul olur, MySQL məlumatları idarə edir, CSS və JavaScript veb səhifənin təqdimatının qayğısına qalır. JavaScript həmçinin veb-serverdə sizin PHP-kodunuz ilə qarşılıqlı təsir gücünə malikdir, nə vaxt ki, nəsə ona yeniləmək lazımdır (necə serverdə, həm də vebstranise-də). Və HTML5-in yeni, yüksək təsirli qalın kətan, audio, video və qeolokasiya kimi xüsusiyyətləri, sizin veb səhifələrinizə daha yüksək dinamikliyi, interaktivliyi və multimediativlik verəcək. indi qısa nəticəni hamıya (hər şeyə) gətirməyə (pis vəziyyətdə qoymağa) yaxşıdır ki, bu başçıda (fəsildə) ifadə edilmişdir, və, proqram kodundan istifadə etmədən, özündə AJAX funksiyasının çox saytlarıyla hər gün istifadə olunanda bu texnologiyalardan bəzilərini uyğunlaşdıran prosesə baxmaq: yeni hesabın qeydiyyatı prosesində yoxlamaya, seçilmiş ad saytın başqa ziyarətçisiylə tutulmamışdırmı (borc alınmamışdırmı). Texnologiyaların belə istifadəsinin yaxşı nümunəsiylə Gmail-in poçt serveri xidmət edə bilər (şəkil. 1.3).



Bu AJAX-proses təxminən növbəti addımlardan ibarətdir.

1. Server lazımlı məlumatları sorğu eləyən veb-formanın yaradılması üçün HTML kodunu verir (ələ verir): istifadəçinin adı, indiki (əsl) ad, indiki (əsl) soyadı və elektron poçt ünvanı.
2. Eyni zamanda bununla server HTML-da istifadəçi adına giriş sahəsinin tərkibini izləməyə və iki şərait yoxlamağa icazə

verən JavaScript-kodu qoyur:

   1. hər hansı mətn bu sahəyə daxil edilmişdirmi;
   2. daxil etmənin fokusu bundan başqa giriş sahəsində istifadəçinin çıqqıltısı üzrə (görə) sahələr yerini dəyişilmiş (köçürdülmüş) idimi.
3. Mətn daxil edilən kimi və daxil etmənin fokusu formanın başqa elementinə yerini dəyişilmişdir (köçürdülmüşdür), fon rejimində JavaScript kodu daxil edilmiş istifadəçinin adını veb-serverdə PHP-ssenariyə verir (xəbər verir) və cavab reaksiyasını gözləyir.
4. Veb-server istifadəçi adına axtarışı həyata keçirir və JavaScript koduna cavabı qaytarır, hansında ki, bildirir, artıq eyni ad cəlb edilmiş idimi.
5. Sonra JavaScript istifadəçi adına giriş sahəsinin altında istifadəçi adına məqbulluq indikatorunu yerləşdirir, ola bilər ki, yaşıl dolaşa və ya qırmızı xaç şəklində, onun mətniylə müşayiət edərək.
6. Əgər istifadəçi qəbuledilməz adı daxil etdisə, amma yenə də formanı göndərməyə çalışır, JavaScript kodu göndərilməni kəsir və başqa adın seçiminin ehtiyacına istifadəçini (mümkün, daha (daha çox) böyük qrafik indikatoru çıxararaq və / və ya xəbərdarlığın pəncərəsini açaraq) diqqət yetirir təkrardır.
7. Bu prosesin təkmilləşdirilmiş versiyası hətta istifadəçi tərəfindən tələb edilmiş adı öyrənə və hal-hazırda alternativ əlçatanı (mümkünü) adı təklif edə bilər.

İstifadəçinin rahatlığı və onlar bütün olanın götürməsinin bütövlüyü üçün bütün bunlar fon rejimində onun diqqətinin cəlb edilməsi olmadan hazırlanır. Serverə AJAX-dan istifadəsiz bütün forma göndəriləcək, sonra o o sahələrin işığıyla HTML kodunu qaytaracaq, hansılarda ki, səhvlər buraxılmışdılar. Olar, əlbəttə, etmək və belə, amma sahənin emalı tələsik daha çox daha maraqlı və daha xoş görünəcək (baxacaq).

AJAX texnologiyası daha çox qərar (həll) üçün məsələlərin geniş mühiti istifadə oluna bilər, nə qədər sadə kontrol və daxil edilən informasiyanın emalı. Sonra bu kitabda AJAX tətbiqiylə reallaşdırılan çox əlavə qəbul baxılacaq.

Bu başçıda (fəsildə) sizin diqqətinizə PHP, MySQL, JavaScript, CSS və HTML5 tətbiqinin əsas texnologiyalarına kifayət qədər tam (dolu) müqəddimə (daxil etmə) təqdim edilmişdi (təsəvvür edilmişdi) (həmçinin Apache) və onların birgə işinin sırası (nizamı) baxılmışdır. 2-ci fəsildə veb-hazırlamalar üçün nəzərdə tutulmuş sizin şəxsi serverinizin quraşdırılmasının üsulları baxılacaq hansında ki, təcrübədə bütün öyrənilən (araşdırılan) materialı öyrənmək mümkün olacaq.

Suallar

1.1 Hansı dörd komponent tamamilə dinamik saytların yaradılması üçün lazımdır?
1.2 HTML abreviaturası Nəyi bildirir?
1.3 Niyə MySQL adında SQL hərfləri olur?
1.4 Və PHP, və JavaScript veb səhifələrin dinamik tərkibini yaradan proqramlaşdırma dilidir. Nədən onların əsas fərqi ibarətdir və niyə siz hər iki bu dil istifadə edəcəksiniz?
1.5 CSS abreviaturası Nəyi bildirir?
1.6 üç əsas yeni HTML5 elementi Sadalayacaqsınız (Keçirdəcəksiniz).
1.7 Əgər siz açıq kodla instrumental vasitələrdən birində səhvi aşkar etməyi bacaracaqsa (ki, nadir hallarda kifayət qədər olur), o necə, sizcə, düzəldilmiş versiyanı almaq olar?
Bu suallara (məsələlərə) cavablar əlavədə (proqramda) A tapmaq olar, "1-ci fəsil məsələlərinə Cavablar" bölməsində.

2

PHP-yə giriş

3-cü fəsildən başlayaraq bu sadə, amma çox güclü dilin öyrənilməsinə başlanacaq və dilin metodikası 7-ci fəsilə qədər aparılacaq.
Mən sizə 2-ci fəsildə qeyd edilmiş birləşmiş işlənmə mühitlərindən (IDE) birində PHP kodunun hazırlamasını yerinə yetirməyə çağırıram. IDE sayəsində kodunuzda olan xətaları aşkar ediləcək ki, bu daha az

funksional redaktorda görülən iş ilə müqayisədə təlimi əhəmiyyətli dərəcədə sürətləndirəcək.
Əksər IDE-lər bu fəsildə göstərilmiş kodları icra edərək çıxış informasiyasını verməyə qabiliyyətlidir. Biz həmçinin HTML faylına PHP kodunun yerləşdirilməsi metodlarına, yəni veb səhifədə çıxış informasiyasının zahiri görünüşü barədə aydınlıq gətirəcəyik.
Biz həmçinin HTML faylına PHP kodunun yerləşdirilməsi metodlarına, yəni veb səhifədə çıxış informasiyasının zahiri görünüşü barədə aydınlıq gətirəcəyik. Veb səhifələrin formalaşma prosesində səhifələr PHP, HTML, JavaScript, CSS, MySQL kombinasiyası vasitəsilə təşkil olunacaq. Həmçinin ola bilər ki, HTML5-in müxtəlif elementlərindən istifadə edilsin. Bundan başqa, hər səhifədə istifadəçiləri istinadlar klikləmək və formaları doldurmaq imkanı verəcək ki, bu da digər səhifələrin çağrılmasına səbəb olacaq. Hərçənd ki, bu dillərin öyrənilməsi zamanı keçinmək (rəftar etmək) olar və belə çətinliklərsiz. Bu mərhələdə PHP-kodun yazılışında və çıxış informasiyasının məzmununun əvvəlcədən söylənilə bilməsi və ya ən azı bu informasiyanın xarakterini anlamaq qabiliyyəti xüsusi olarq oxuculara aşılanacaq

HTML-ə PHP-nin əlavə edilməsi

PHP-sənədlərin ad sonluğu susmaya görə PHP genişlənilməsi ilə qoyulur. Veb-server sorğu edilən faylda bu genişlənmə ilə rastlaşdıqda, sözügedən server avtomatik olaraq faylı PHP-prosessora avtomatik ötürür. Veb-serverlər qurmaların (kökləmələrin) kifayət qədər geniş diapazonuna malikdir, və bəzi veb-istehsalçılar belə iş rejimini seçirlər ki, təhlil (götürmə) üçün PHP-prosessora məcburidir həmçinin HTM və ya HTML genişlənilmələriylə faylları ötürülürlər. Adətən bu onunla bağlıdır ki, istehsalçılar PHP-dən istifadə faktını gizlətmək istəyirlər.

PHP-də proqram təmiz halda faylın qayıdışı üçün cavab verir. Ən sadə hadisədə PHP sənədinin çıxışında yalnız HTML kodu alınacaq. Bundan əmin olmaq üçün, HTML sənədi tərəfindən götürmək olar, məsələn index.html faylı, onu index.php adının altında saxlamaq, və o əks olunacaq dəqiq kimi ilk fayl.

PHP komandalarının icrası üçün yeni teq öyrənmək lazımdır. Bu teqin açılışı belədir:

```
<?php
```

Birinci php, nə gözlərə atıla bilər, — teqin başa çatdırılmaması. Teqin daxilində tam olaraq PHP kodunun fraqmentləri yerləşdirilə bilər. Bu

teq,

```
?>
```

ilə bağlanılır.

Nümunə 3.1-də kiçik Hello World proqramının PHP dilində yazılmış növü göstərilmişdir. Nümunə 3.1. PHP-nin çağırışı

```
<?php
echo "Hello world";
?>;
```

Bu teqin istifadəsi çox elastikdir. Bəzi proqramçılar bu teqi sənədin başlanğıcında açır, sənədin sonunda isə bağlayır və PHP komandalarının bilavasitə istifadəsi ilə istənilən HTML kodundan istifadə edə bilirlər.

Başqa növ proqramçılar isə bu teqlərdən yalnız dinamik ssenariləri tətbiq etdiyi yerlərdə yerləşdirməyə üstünlük verirlər. Bu da digər üsul ilə daha kiçik və yığcam PHP kodların alınması ilə nəticələnir.

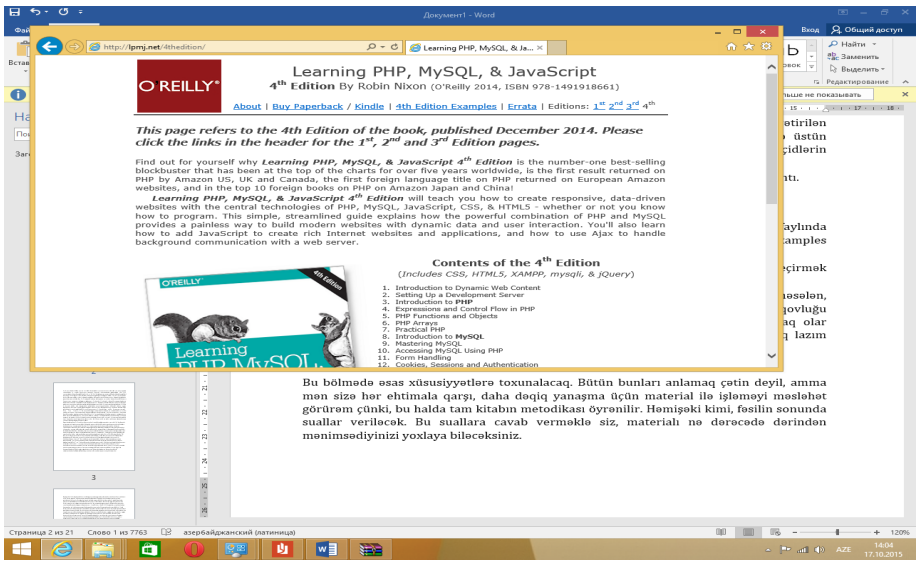
Sonuncu proqramlaşdırma metodunun tərəfdarları öz seçimini adətən onunla əsaslandırırlar ki, bu cür kodlar daha sürətli yerinə yetirilir. Birinci metodun tərəfdarları iddia edirlər ki, sürətin artımı o qədər cüzidir ki, bu artım, sənəddə ayrı-ayrı götürülmüş çoxsaylı PHP kodlarının yerləşdirməsində yaranan əlavə çətinliklərə bəraət qazandıra bilməz.

Dilin öyrənilməsi müddətində siz, şübhəsiz ki, PHP-da, amma bu kitabda gətirilən nümunələrin sadələşdirilməsi üçün hazırlamaların yaradılması vaxtı öz üslub üstün tutmalarında müəyyən edin mən minimuma PHP-ın və HTML-ın arasında keçidlərin miqdarını yaxınlaşdırdım, orta hesabla bir-ikiyə bir sənədə keçidlərə.

Yeri gəlmişkən, mövcuddur və PHP sintaksisinin bir neçə (bir qədər) başqa variantı.

Bu kitabdakı nümunələr

Siz bu kitabda göstərilən nümunələrin məcmusu 2nd_ edition_examples.zip arxiv faylında yerləşdirilmiş. Bu faylı yükləmək üçün saytda daxil olaraq Download Examples (Nümunələri yükləmək) istinadında klikləməyiniz kifayətdir. (şəkil. 3.1).



Şəkil. 3.1. Bu kitabda göstərilən nümunələri http://lpmj.net ünvanında gözdən keçirmək olar

Bütün nümunələr nömrələnmişdir və fəsillərə uyğun qruplaşdırılmışdır (məsələn, example3-1.php). Bundan başqa, saytda həmçinin əlavə olaraq named_examples qovluğu var ki, burada da konkret adlarla saxlanmış nümunə fayllarını burada tapmaq olar (məsələn nümunə 3.4-də göstərilmiş nümunəni, test1.php adlı faylda saxlamaq lazım olacaq)

PHP strukturu

Bu bölmədə əsas xüsusiyyətlərə toxunalacaq. Bütün bunları anlamaq çətin deyil, amma mən sizə hər ehtimala qarşı, daha dəqiq yanaşma üçün material ilə işləməyi məsləhət görürəm çünki, bu halda tam kitabın metodikası öyrənilir. Həmişəki kimi, fəsilin sonunda suallar veriləcək. Bu suallara cavab verməklə siz, materialı nə dərəcədə dərindən mənimsədiyinizi yoxlaya biləcəksiniz.

Şərhlər

PHP koduna şərhlərin əlavə edilməsinin iki üsulu mövcuddur. Birincisi, şərh sətirinin başlanğıcında iki düz sləş yerləşdirməklə:

```
//Bu şərhdir
```

Bu tip şərhlər, mənbə kodunda olan xətalı kod sətirinin proqramdan müvəqqəti çıxarılması üçün də istifadə edilə bilər. Məsələn, izah

etmənin belə üsulundan, ehtiyac yaranmadığı halda kod sətirinin gizlətmək üçün tətbiq edilə bilər:

```
//echo "X equals $x";
```

Belə şərhi kodun gördüyü işi təsvir etmək üçün kod sətirindən sonra da yerləşdirmək olar:

```
$x += 10; //$x qiymətinin 10-dan artımı
```

Əgər şərhiniz bir neçə sətirdən ibarətdirsə, onda izah etmənin ikinci üsulundan istifadə etmək lazımdır, hansı ki, bu üsul nümunə 3.2-də göstərilmişdir.

Nümunə 3.2. Multixəttli şərh

```
<?php /* Bu multixətt şərhin sahəsidir,
hansı ki, interpretasiyaya
məruz qalmayacaq */
```

```
? >
```

Praktik olaraq kodun istənilən sərbəst seçilmiş yerində şərhin açılışı və bağlanması üçün / və */ simvollarından istifadə etmək olar . Şərhlərin yerləşdirilməsi zamanı buraxılan tipik xəta — kommentari verilmiş sahədə olan mətnin əlavə olaraq */ və /* simvollarını ehtiva etməsidir. Şərhlər bir-birilə uyğunlaşdırıla bilmir, çünki, PHP-şərhçi harada şərhin qurtaracağını anlamır və xətanın olduğu barədə məlumat bildirir. Amma əgər siz proqramlaşdırma üçün uyğunlaşdırılan və ya sintaksis işıqlanma ilə təchiz edilmiş redaktorlardan istifadə edirsinizsə, onda bu cür xətaları aşkar etməyiniz asandır.*

Əsas sintaksis

PHP çox elastik dildir, amma bu dilin sintaksisinə və strukturuna aid olan bir neçə qayda mövcuddur ki, onları öyrənmək lazımdır.

Nöqtəli vergül

Əvvəlki nümunələrdə göstərildiyi kimi PHP komandaları nöqtəli vergül ilə bitir:

```
$x += 10;
```

PHP ilə işləyərkən əksər hallarda nöqtəli vergülləri qoymağı unutmağınız mümkündür. Nöqtəli vergülü qoymadığınız halda, PHP bir neçə təlimat eyni anda baxmalıdır, çünki nöqtəli vergüllər təlimatları bir-birindən ayırır. Nöqtəli vergüllərin qoyulmaması halında isə, PHP vəziyyəti anlaya bilməyəcək və kodun sintaktis təhlili zamanı Parse error xətasının olduğunu göstərək.

$ simvolu

$ simvolu müxtəlif proqramlaşdırma dillərində müxtəlif məqsədlər üçün istifadə olunur. Məsələn, BASIC dilində $ simvolu dəyişən adlarının tamamlanmasında tətbiq edilirdi, göstərmək üçün ki, onlar sətirlərə aiddirlər.

PHP-də $ simvolu bütün növ dəyişənlərin adlarından əvvəl qoyulmalıdır. Müxtəlif tiplərə uyğun dəyişənlər nümunə 3.3-də göstərilmişdir.

Nümunə 3.3. Dəyişənlərə üç məlumat tipində olan qiymətlərin mənimsədilməsi

```
<?php
$mycounter = 1;
$mystring = "Hello";
$myarray = array ("One", "Two", "Three");
? >
```

Bu tamamilə doğru sintaksis formasıdır və məsləhətdir ki, bu cür yazılış formasını mənimsəyəsiniz. Başqa dillərdən fərqli olaraq, PHP-də proqramın mətnində istənilən qədər boşluq buraxa bilərsiniz və PHP bu boşluqlara etinadsız yanaşır. Həqiqətdə bu, dilin səmərəli istifadəsi ki, boş sahə adlanır, adətən həvəsləndirilir (hərtərəfli izah etməylə yanaşı), bir halda ki, (çünki,) şəxsi kodu anlamağa kömək edir, nə vaxt ki, ona bəzi (bir qədər) vaxt keçəndən sonra qayıtmaq lazım olur. Bu kömək edir və sizin kodunuzu dəstəkləməyə məcbur edilmiş başqa proqramçılara. Əgər faylda yalnız PHP kodu, onda bu kodu ?> teqi ilə bağlamaq olar. Həm də PHP fayllarında artıq boş yerin qalmaması üçün (xüsusilə, obyekt yönümlü kodun yazılışı zamanı) bu üsuldan istifadə etmək məsləhətdir.

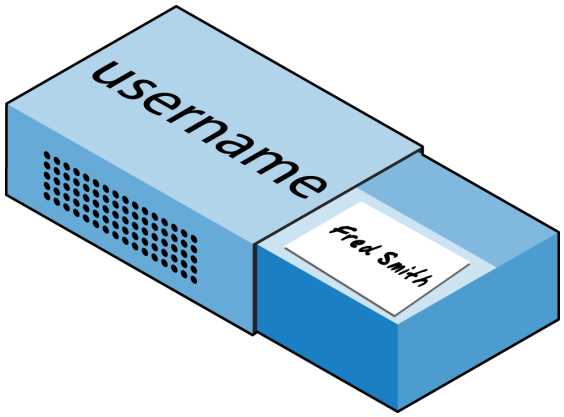
Dəyişənlər

PHP dəyişənlərinin nə olduğunu anlamağınızda, sizə sadə metafora kömək edəcək. Kiçik (və ya böyük) kibrit qutuları barədə düşünün! Rənglənmiş və adları yazılmış kibrit qutularına fikir verin.

Sətir tipində olan dəyişənlər

Təsəvvür edin ki, sizdə üzərində username (istifadəçinin adı) sözü yazılmış qutu mövcuddur. Sonra siz kağız parçasına Fred Smith yazıb və bu kağızı qutuya qoyursunuz. Bu proses dəyişənə sətir tipində olan qiymətin mənimsədilməsinə oxşayır:

```
$username = "Fred Smith";
```



Şəkil. 3.2. Dəyişənlər, predmetlərdən ibarət olan bir kibrit qutusu şəklində təqdim etmək olar

Dırnaq əlaməti ona bildirir ki, Fred Smith simvol birləşməsi sətir tipindədir. İstənilən sətirin ikiqat və ya tək dırnaqlar (apostroflar) daxilində olması vacibdir. Bu iki dırnaq növü arasında olduqca əhəmiyyətli fərq mövcuddur (bu fərqlər sonrakı bölmələrdə izah ediləcək).

Qutunun daxilində olan kağıza baxmaq üçün, kağızı qutudan çıxardırsınız və kağızda nə yazıldığını oxuyursunuz. PHP-də belə hərəkət buna bənzəyir:

```
echo $username;
```

Həmçinin başqa dəyişənin tərkibinə də mənimsətmə (kağızın kserokopiyasını etmək və onu başqa qutuya yerləşdirmək) mümkündür:

```
$current_user = $username;
```

Əgər nümunələri birləşmiş işləmə mühitində (2-ci fəsilin sonunda verilmiş tövsiyələr) yoxlamağı cəhd edə bilərsiniz. Elə indi aşağıdakı nümunədə göstərilən PHP kodu hər hansı mətn redaktoruna kopyalayıb, hazırladığınız sənədi serverinizin ilk kataloqunda test1.php adlı fayl kimi saxlayın.

Nümunə 3.4. Sizin ilk PHP-proqramınız

```
<?php
//test1.php
$username = "Fred Smith";
echo $username;
echo "<br>";
current_user = $username;
echo $current_user;
? >
```

İndi isə brauzerin ünvan sətirinə aşağıdakı ünvanı daxil edib bu proqramı icra etmək olar:

```
http://localhost/test1.php
```

Əgər veb-serverin quraşdırılması (2-ci fəsildə göstərilmişdir) gedişatında siz serverə təyin edilmiş portu 80-dən fərqli hər hansı başqa bir port nömrəsi ilə dəyişdirmisinizsə, bu halda bu və kitabda yerləşdirilmiş bütün nümunələrdə URL ünvan sahəsinə həmçinin daxil etdiyiniz port nömrəsini də yerləşdirməlisiniz. Məsələn, əgər siz portu 8080 nömrəli porta dəyişdirmisinizsə, onda əvvəlki URL aşağıdakı şəkildə olacaq:

```
http://localhost:8080/test1.php
```

kitabdakı digər nümunələrin yoxlanması və ya şəxsi kodun yazılışı zamanı bu xatırlatmanı unutmayın.

Kodun icrası zamanı iki dəfə Fred Smith adı yazılacaq, birinci yazılışın səbəbi — echo $username komandasının icrası, ikinci yazılışın səbəbi isə — echo $current_user komandasının icrasıdır.

Ədəd dəyişənləri

Dəyişənlər sətirlərdən başqa ədəd məlumat tipində olan qiymətləri özündə ehtiva edə bilər. Məsələn, $count dəyişəninə 17 ədədi mənimsədilib. Əgər analoji olaraq kibrit qutusu bənzətməsinə qayıtsaq, onda üzərində $count sözü yazılmış qutuda 17 muncuq dənəsi mövcuddur:

```
$count = 17;
```

dəyişənə həmçinin kəsr ədədləri mənimsətmək olar və sintaksis əvvəlki qaydadır:

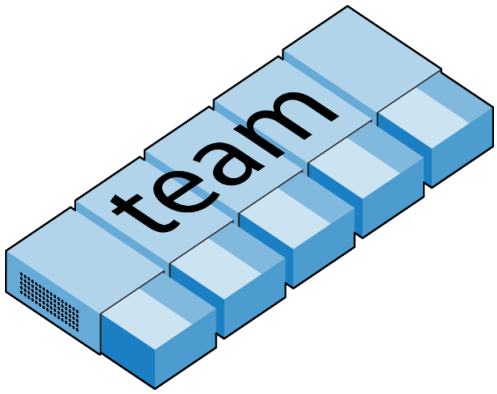
```
$count = 17.5;
```

qutunun tərkibi haqqında bilmək üçün, əlbəttə ki, qutunu açmaq və muncuq dənələrini saymaq lazımdır. PHP-də $count dəyişəninin qiymətini başqa bir dəyişənənə mənimsətmək və ya echo komandasından istifadə edib bu dəyişənin qiymətini brauzer vasitəsilə ekranlaşdırmaq olar.

Massivlər

Bir-birinə yapışdırılmış kibrit qutularının birləşməsini massiv kimi təsəvvür etmək olar. Məsələn, bizə bir komandanın beş futbolçusunun adını $team massivində saxlamaq lazımdır. Bunun üçün biz beş dənə qutudakı hər bir kağıza göstərilən futbolçuların adını yazıb, kağızı öz qutusuna qoyub, qutuları yan-yana yapıştırmalıyıq. Birlikdə yapışdırılmış qutuların üzərinə team sözünü yazacağıq (şəkil. 3.3). PHP-də bu hadisəyə ekvivalent kod aşağıdakı şəkildə olacaq:

```
$team = array ('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```



Şəkil. 3.3. Massiv, bir-birinə yapışdırılmış bir neçə kibrit qutusuna bənzəyir

Bu sintaksis daha əvvəl baxılmış təlimatlardan bir qədər daha mürəkkəbdir. Massiv, aşağıdakı konstruksiya vasitəsilə yaradılır: array(); yumru mötərizələrin daxilində beş sətir göstərilmiş və bu sətirlər massivin elementləridir. Hər bir sətir tək dırnaqlar ilə əhatələnmişdir. 4-cü oyunçunun kim olduğunu bilmək istəyirsinizsə, aşağıdakı komandadan istifadə etmək lazımdır:

```
echo $team[3]; //Bu komanda Chris adını göstərəcək.
```

Deyə bilərsiniz ki, bizə 4-cü futbolçu lazım idi, amma kvadrat mötərizələrin daxilində 3 rəqəmi daxil edilmişdir. Bunun səbəbi odur ki, PHP-də massiv elementləri sıfırdan başlayaraq indeksləşdirilir. Elə buna görə də əvvəlki nümunədəki massivin beş elementi [0-4] aralığında indeksləşdirilir və massivin elementlərindən istifadə edilərkən, bu aralıqdan kənara çıxmaq olmaz.

İkiölçülü massivlər

Massivlərdən istifadənin diapazonu çox genişdir. Məsələn, onlardan bir sıra qutu düzülüşünün birölçülü forması yerinə ikiölçülü, üçölçülü və daha çox ölçülü matris qurmaq olar. İkiölçülü massivə ən yaxşı nümunə üçün X-0 (iks-nolik) oyunun gedişini izləmək lazımdır. Bu zaman görəcəksiniz ki, 3 × 3 sahəli kvadrat daxilində qruplaşdırılmış doqquz qəfəs strukturu tələb olunur. Bunu kibrit qutuları şəklində təqdim etmək üçün, matris şəklində yapışdırılmış üç sətirdən və üç sütundan ibarət olan doqquz qutu təsəvvür edin (şəkil. 3.4).

Şəkil. 3.4. Qutuların köməyi ilə hazırlanmış çoxölçülü massivin modeli

İndi hər gediş üçün lazım olan qutulara X və ya 0 yazılmış kağız parçalarını qoymaq olar.
Bunu PHP-də etmək üçün nümunə 3.5-də göstərildiyi kimi, üç müxtəlif massivi özündə ehtiva edən massiv yaratmaq lazımdır:

Nümunə 3.5. İkiölçülü massivin təyini

```
<?php
$oxo = array (array ('x', ' ', 'o'), array ('o', 'o', 'x'), array ('x',
'o', ' '));
?>
```

Biz hal-hazırda çətin bir mərhələni keçdik, amma məzmun sadə şəkildə oxucuya çatdırılmışdır və ümid edirəm ki, əgər massivin əsas sintaksisini mənimsəmisinizsə, bu mərhələni tam anlayacaqsınız. Burada üç array() konstruksiyası bir array() konstruksiyasına qoşulmuşdur. Bu massivin elementlərindən istifadə etmək üçün məsələn, əsas massivin ikinci sətirində yerləşmiş massivin üçüncü elementindən aşağıdakı PHP-komandası vasitəsilə istifadə etmək olar. Qeyd edək ki, nümunədə "x" simvolu ekranlaşdırılacaq:

```
echo $oxo [1] [2];
```

Bir daha qeyd edək ki, massivin indekslərinin (massivin daxilindəki elementlərə göstəricilər) hesablanması vahiddən deyil, sıfırdan başlanır, buna görə də əvvəlki komandadakı [1] indeksi üç elementli massivin ikinci

elementinə (nümunədə bu element massivdir) istinad edir. [2] indeksi isə — massivdaxili ikinci massivin üçüncü elementinə istinad edir.

Artıq deyildiyi kimi, hətta bir-birini böyük miqdarda qoyulmuşun sadə yaradılması yolu ilə massivlər alınan ən böyük ölçüylə massivlər dəstəklənir. Amma bu kitabda daha çox ikiölçülü massivlər baxılmayacaq. Massivlər haqqında biz 6-cı fəsildə danışacağıq.

PHP-dəyişənlərin yaradılması

Dəyişənlərə adların mənimsədilməsinin dörd qaydası mövcuddur.

- Dəyişən adları hərflər və ya _ (sətir xətti) simvolu ilə başlanmalıdır.
- Dəyişən adları yalnız bu simvollardan ehtiva oluna bilər: a – z, A – Z, 0 – 9 və _ (sətir).
- Dəyişən adları boşluqlardan ehtiva etməməlidir. Əgər dəyişənin bir neçə sözdən ibarətdirsə, onda ayırıcı simvol kimi sətir xəttindən (məsələn, $user_name) simvolundan istifadə etmək lazımdır.
- Dəyişən adlarının simvolları registrə həssasdır. $High_Score dəyişəni $high_score dəyişənindən fərqlənir.

PHP-də diakritik işarələrdən ibarət ASCII-simvollarından istifadəyə zəmin yaratmaq üçün dəyişən adları 127-255 baytlıq simvolları dəstəkləyir. Amma hələlik kodunuzda belə simvolların tətbiqindən çəkinin, çünki, klaviaturanın ingilis düzülüşlərindən istifadə edən proqramçılar belə simvolların daxil edilməsi zamanı çətinlik çəkirlər.

Operatorlar

Operatorlar — riyazi (məsələn, "üstəgəl" kimi, "çıxma", "vurma" və "bölmə" kimi), sətir, məntiqi və müqayisə komandalarıdır. PHP kodu, adi hesab əməliyyatlarına bənzəyir. Məsələn, aşağıdakı operatorun işi nəticəsində 8 rəqəmi ekranlaşdırılır:

```
echo 6 + 2;
```

PHP imkanlarının öyrənilməsinə başlamazdan əvvəl, dildə olan müxtəlif operatorların verdiyi imkanlar barədə diskusiya aparmaq lazımdır. Hesab operatorları Hesab operatorları adından məlum olduğu kimi tamamilə hesab əməliyyatını edirlər. Bu növ operatorlar, riyazi əməliyyatların icra edilməsi üçün tətbiq edilirlər. Bu operatorlardan dörd əsas (toplama, çıxma, vurma və bölmə), həmçinin qalıq və bir vahid artım və ya azalmanın tapılması əməliyyatının həyata keçirilməsi üçün istifadə etmək olar (cədvəl. 3.1).

Mənimsəmə operatorları

Bu operatorlardan, dəyişənlərə qiymətlərin mənimsədilməsi üçün istifadə olunur. Onlardan ən sadəsi = operatorudur. Həmçinin +=, - = və s. operatorlar da mövcuddur. (ətraflı cədvəl. 3.2). += operatoru solda olan qiymətin tam əvəzlənməsinin yerinə bu qiymətə sağda olan qiyməti əlavə edir. Beləliklə, əgər $count dəyişənin ilkin qiyməti 5-dirsə $count += 1 ifadəsi ilə $count dəyişəni 6 qiymətinə bərabər olur. Bu ifadə adət edilmiş operatoru mənimsəmə operatoru ilə qurulmuş bu ifadəyə ekvivalentdir:

```
$count = $count + 1;
```

Cədvəl 3.2. Mənimsəmə operatorları

Operator	Nümunə	Ekvivalenti
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 5	$j = $j - 5
*=	$j *= 5	$j = $j * 5
/=	$j /= 5	$j = $j / 5
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Sətirlərdə nöqtə (.) şəxsi operatoru var ki, bu barədə daha ətraflı "Sətirlərin bitişdirilməsi" bölməsində baxılacaq.

Müqayisə operatorları

Bir qayda olaraq, müqayisə operatorları iki elementin qiymətlərini müqayisə etmək tələb olunan konstruksiyalar daxilində (if təlimatı kimi) istifadə olunur. Məsələn, əgər dəyişənin artıma məruz qaldığını yoxlamaq, iki və daha artıq dəyişənin qiymətlərinin müqayisə edilməsi və s. bu kimi əməliyyatlarda bu operatorlardan istifadə olunur (cədvəl. 3.3).

Cədvəl 3.3. Müqayisənin operatorları

Operator	Mənası	Nümunə
==	Bərabərdir	$j == 15
!=	Bərabər deyil	$j != 21
>	$j -= 5	$j > 5
<	$j *= 5	$j < 100
>=	$j /= 5	$j >= 17
<=	$j .= $k	$j <= 8

Nəzərə alın ki, = və == operatorları müxtəlif təsirlər üçün nəzərdə tutulmuşdur. Birinci operator mənimsəmə operatoru olduğu halda, ikinci operator müqayisə operatorudur. Bəzən hətta təcrübəli proqramçılar belə, bu operatorları səhv salırlar, ona görə də bu operatorlardan istifadə edərkən diqqətli olun.

Məntiqi operatorlar

Əgər məntiqi operatorlar ilk dəfədir tanış olursunuzsa, onda bu operatorlar əvvəlcə sizə qeyri-adi görünə bilər. Sadə danışıq dilində də məntiqi nəticələrdən istifadə etdiyinizi təsəvvür edin. Məsələn: "Əgər saat 12-dən çox, 14-dən azdırsa, deməli, nahar vaxtıdır". PHP-də bu fikiri bu cür alqoritmləşdirmək olar:

```
if ($hour>12 && $hour<14) dolunch();
```

Burada nahar, gələcəkdə yaradılması mümkün olan və gördüyünüz təlimatlar dəstinə malik dolunch adlı funksiya şəklində yerləşdirilmişdir.

Əvvəlki nümunədən göründüyü kimi, məntiqi operator adətən əvvəlki bölmədə göstərilmiş iki operator arasında müqayisə əməliyyatının nəticələrinin birləşdirilməsi üçün istifadə olunur. Bir məntiqi operatorun işinin nəticəsi başqa bir məntiqi operator üçün giriş

qiyməti ola bilər (" Əgər saat 12-dən çox, 14-dən azdırsa və ya əgər otaqdan qızartma qoxusu gəlirsə və masanın üzərində boşqablar düzülmüşsə..."). Bir qayda olaraq, əgər hansısa bir əməliyyat həqiqi və ya yanlış qiymətə malikdirsə — bu hərəkətlər TRUE və ya FALSE qiymətlərini alır. Məntiqi operatorlar aşağıdakı cədvəldə göstərilmişdir.

Cədvəl 3.4. Məntiqi operatorlar

Operator	Mənası	Nümunə
&&	VƏ	$j ==5 && $k==3
and	Aşağı prioritetli VƏ	$j ==5 and $k==3
||	VƏ YA	$j < 6 || $k > 9
or	Aşağı prioritetli VƏ YA	$j < 6 || $k > 9
!	DEYİL	! ($j == $k)
xor	VƏ YA çıxarışı	$j xor $k

Qeyd edək ki, && operatoru adətən and operatoru ilə qarşılıqlı əvəz oluna bilər; həminki qayda ilə || operatoru or operatoru ilə əvəzlənə biləq. Amma bir məsələ də var ki, and və or operatorlarından oxşarlarından aşağı prioritetlidi, buna görə də bəzi hallarda prioritetləri yerbəyer etmək üçün əlavə mötərizələrdən istifadə edilir. Eyni zamanda yalnızca and və ya or operatorlarından da istifadə olunur:

```
mysql_select_db ($database) or die ("Məlumat bazasını seçmək mümkün deyil");
```

Bu operatorlardan sizə *VƏ YA-nın çıxarışı* olan xor operatoru tanış olmaya bilər. Bu operator, yalnız o halda icraya başlayır ki, operatorda qeyd edilmiş hər bir dəyişənin qiyməti ya TRUE ya da FALSE olsun.

Bu əməliyyatı anlamaq üçün təsəvvür edin ki, siz təmizlik üçün nəzərdə tutulan vasitə ixtira etmək istəyirsiniz. Həm ammonyak (ammonia), həm də xlorlu əhəng (bleach) yaxşı təmizləmə xüsusiyyətlərinə malikdir. Lakin vasitənizin tərkibində bu maddələrdən yalnızca biri olmalıdır. Çünki onların ikisi də bir tərkibdə ola bilməz, çünki, bu maddələrin bir-birilə təması təhlükəlidir. PHP-də bu hadisəni aşağıdakı qaydada təqdim etmək olar:

```
$ingredient = $ammonia xor $bleach;
```

Təqdim edilmiş fraqmentdə, iki dəyişəndən - $ammonia və ya $bleach-dən hər hansı biri, TRUE qiymətini alarsa, onda $ingredient dəyişəninin qiyməti də TRUE təyin ediləcək. Amma əgər dəyişənlərin

hər ikisi TRUE və ya FALSE qiymətini, onda $ingredient dəyişəninin qiyməti FALSE təyin ediləcək.

Dəyişənlərə qiymətlərin mənimsədilməsi

Dəyişənin qiymətin mənimsədilməsi sintaksisi bu formadadır: *dəyişən = qiymət*. Bir dəyişən qiymətinin başqa bir dəyişənin qiymətinə ötürülməsi bu formadadır — *başqa_dəyişən = dəyişən*.

Digər mənimsəmə operatorları da mövcuddur ki, işinizdə faydalı ola bilər. Məsələn, artıq bu operator bizə tanışdır:

```
$x += 10;
```

Bu operator, PHP-parserə əmr edir ki, $x dəyişəninin qiymətinə ondan sağda olan (indiki halda bu qiymət 10-a bərabər) qiyməti əlavə etsin. Oxşar tərzdə də çıxma əməliyyatını yerinə yetirmək olar:

```
$y - = 10;
```

Dəyişənin qiymətinin bir vahid artırılması və azaldılması

Vahidin əlavə edilməsi və ya vahidin çıxılması — o qədər sıx istifadə edilən əməliyyatdır ki, PHP bu əməliyyat üçün xüsusi operatorlar təyin edib. += və - = operatorlarının yerinə aşağıdakı operatorlardan birini istifadə etmək olar:

```
++$x;
--$y;
```

Operatorları yoxlamaq üçün aşağıdakı koddan istifadə etmək olar:

```
if (++$x == 10) echo $x;
```

Bu kod PHP-yə əvvəlcə $x dəyişəninin qiymətini vahidə artırmaq əmri verir və alınan qiymətin 10 olduğunu yoxlayır; əgər dəyişən bu qiymətə malikdirsə, onda bu dəyişəni ekranlaşdırmaq lazımdır. PHP-də həmçinin dəyişənin qiymətinin vahid azalmasını tətbiq etmək olar və yuxarıdakı nümunəyə analoji olaraq, bu əməliyyat aşağıdakı kodda göstərilmişdir:

```
if ($y-- == 0) echo $y;
```

Bu kodlar fərqli nəticələr verir. Fərz edək ki, operatorun icrasına qədər $y dəyişəninin ilkin qiyməti sıfıra bərabər idi. Müqayisə əməliyyatı TRUE nəticəsini alacaq, amma bundan sonra, $y dəyişəninə −1 qiyməti mənimsədiləcək. Bəs onda echo təlimatı hansı qiyməti əks etdirəcək: 0 ya −1? Bunu təxmin etməyə çalışın, və öz təxmininizi təsdiq

etmək üçün, PHP-prosessorda təlimatın işini sınaqdan keçirin. Çünki, operatorların belə kombinasiyası sizi nəzəri cəhətdən çaşdıra bilər və məsləhətdir ki, nümunəni praktiki edin. Bu yazılışa heç bir halda münasib proqramlaşdırma stili kimi baxmaq olmaz. Sözün qısası, məhz nə vaxt artırılmışdır və ya dəyişənin qiyməti (mənası) vahidə azaldılmışdır, qədər və ya yoxlamadan sonra, ondan asılıdır, harada artımın və ya dekrementin operatoru yerləşdirilmişdir — dəyişənin adından əvvəl (qarşısında) və ya ondan sonra. Yeri gəlmişkən, əvvəlki sualın doğru cavabı: echo təlimatı −1 nəticəsini əks etdirəcək, çünki $y dəyişəninin qiyməti dərhal vahidə ondan sonra azaldılmışdı, necə ona if təlimatı giriş əldə etdi, və ona qədər, necə ona echo təlimatı giriş əldə etdi.

Sətirlərin bitişdirilməsi

Sətirlərin bitişdirilməsi zamanı, bir sətir simvolları başqa bir sətir simvollarına əlavə edilir. PHP-də bu əməliyyat nöqtənin simvolu (.) vasitəsilə həyata keçirilir. Sətirlərin bitişdirilməsi ən sadə üsulu buna bənzəyir:

```
echo "Sizin". $msgs. "mesajınız var.";
```

Fərz etsək ki, $msgs dəyişəninə 5 qiyməti mənimsədilmişdir, o zaman bu kod aşağıdakı sətiri ekranlaşdıracaq:

```
Sizin 5 mesajınız var.
```

Bildiyimiz kimi +=operatorun köməyi ilə ədəd dəyişəninə qiymət əlavə etmək olar. Analoji qaydanı sətirlərdə .= operatorunun köməyilə etmək olar:

```
$bulletin .= $newsflash;
```

İndiki halda, əgər $bulletin dəyişənində xəbərlər barədə məlumat əlavə edilərsə, yeni və fövqəladə xəbərlər kimi fərqləndirilən $newsflash dəyişəninə mənimsədilən qiymət həmçinin də $bulletin dəyişəninə əlavə ediləcək və belə olan halda $bulletin dəyişəni hər iki sətiri ehtiva edəcək.

Sətir tipi

PHP-də iki sətir tipi dəstəklənir ki, bunlar dırnaq tipinə fərqlənir. Əgər dəyişənə mətn sətirin tərkibini olduğu kimi saxlamaqla mənimsətmək tələb olunursa, onda tək dırnaqlardan istifadə etmək lazımdır:

```
$info = 'Dəyişən adları $ simvolu vasitəsi yazılır və yazılış qaydası bu
nümunədə göstərilmişdir: $variable';
```

Indiki halda $info dəyişənindəki tək dırnaqlar sətirin daxilində olan hər bir simvola mənimsədilir. Əgər bu dəyişəndə ikiqat dırnaqdan istifadə edilsə, onda PHP $variable dəyişənini hesablamağa və qiymət almağa çalışacaq. Eyni zamanda, əgər sətirin tərkibində dəyişən qiymətini daxil etmək tələb olunursa, ikiqat dırnaqlara əhatələnmiş sətirdən istifadə oluna bilər:

```
echo "Bu həftə ərzində sizin profilinizə $count istifadəçi baxmışdır";
```

Bu bu sintaksis, "Sətirlərin bitişdirilməsi" bölməsində göstərilən formadan daha sadədir, hansı ki, sətirə başqa bir sətirin əlavə edilməsi üçün nöqtə simvolundan istifadə edərkən tək dırnağı bağlayıb, mətnə dəyişəni əlavə etdikdən sonra tək dırnağı yenidən açmağa gərək qalmır. Bu qəbul forması, dəyişənin əvəz edilməsi adlanır. Qeyd etmək olar ki, bəzi proqramlarda bu üsula kifayət qədər tez-tez rast gəlinir, amma bəzi proqramlarda isə, ümumiyyətlə tətbiq edilmir.

Simvolların təyinatının dəyişikliyi

Bəzən elə hallar olur ki, sətirdə xüsusi təyinata malik simvolları yerləşdirmək lazımdır və bu simvolların pərakəndə tətbiqi sintaksis səhvə səbəb olur. Məsələn, aşağıdakı kod sətiri işləməyəcək, çünki mətnin ikinci sözündə - spelling's sözündə qoyulmuş apostrof səbəbilə, PHP-parser elə zənn edəcək ki, bu sözdə qoyulmuş tək dırnaq mətnin sonunu bildirir. Beləliklə, sətirin apostrofdan sonrakı hissəsi səhv kimi qeyd ediləcək:

```
$text = 'My spelling's atroshus'; //Səhvin düzəldilməsi
```

Dırnaq xətasını tək dırnağın əks sləş simvolu vasitəsilə çağrılan çoxmənalı simvol aləti ilə düzəltmək olar. Bunun üçün əks sləşi () tətbiq etmək istədiyimiz xüsusi simvoldan (indiki halda bu simvol: ') qabaq daxil etmək lazımdır. Bu zaman PHP əks sləş ilə tətbiq edilən simvola sətir kimi baxacaq və simvol interpretasiyasına məhəl qoymayacaq:

$text = 'My spelling's still atroshus';

Bu qəbul formasında bütün vəziyyətlərdə (yəni, digər xüsusi simvollarda) praktik olaraq tətbiq etmək olar. Məsələn, ikiqat dırnaqlar ilə əhatələnmiş aşağıdakı sətir dəyişənə səhvsiz mənimsədiləcək:

$text = "She wrote upon it, " Return to sender".";

Bundan başqa, sətirə müxtəlif xüsusi simvolların məsələn tabulyasiya, yeni sətir və karetin alınmasını yerləşdirmək üçün, idarəedici simvollardan istifadə olunur: \t, \n və \r (sadalanlara

müvafiq olaraq göstərilmişdir). Başlıq nişanı üçün istifadə edilən tabulyasiya simvollarının tətbiqi aşağıdakı nümunədə göstərilmişdir

```
$heading = "Gün\tAd\tPlatej";
```

Əks sleşa simvolları vasitəsilə daxil edilən xüsusi simvollar yalnız ikiqat dırnaq ilə əhatələnmiş sətirlərdə dəstəklənir. Əgər əvvəlki sətirdə tək dırnaqdan istifadə etsək, onda mətndə tabulyasiya simvollarının yerinə heç bir mənası olmayan \t simvol ardıcıllıqları əks etdiriləcək. Tək dırnaqlara əhatələnmiş sətirlərin daxilində əks sləş simvolları ilə yalnız təyinatı dəyişdirilmiş simvollar (apostrof (') və əks sləşin özü (\)) dəstəklənir.

Multixətt komandalar

Bəzən elə olur ki, PHP-də böyük həcmli mətnin daxil edilməsinə ehtiyac yaranır, lakin bu mətnin ekranlaşdırılması üçün echo və digər bu tip bir neçə təlimatdan (məs. print) istifadə çox vaxt aparır və bu təlimatlar ilə işin gedişatı məntiqsizdir. PHP, belə vəziyyətlərin öhdəsindən gəlmək üçün nəzərdə tutulmuş iki rahat vasitə təklif edir. Onlardan birincisi bir neçə sətirin dırnaqların nəticədən ibarətdir, necə nümunə 3.6-da. Həmçinin dəyişənlərə qiymətləri nümunə 3.7-də göstərilmiş üsul ilə də mənimsətmək olar.

Nümunə 3.6. *echo təlimatı bir neçə sətirdə istifadə edilir*

```
<?php

$author = "Steve Ballmer";

echo "Developers, Developers, developers, developers, developers, developers, developers, developers, developers! - $author.";

?>
```

Nümunə 3.7. *Multixətt mənimsəmə*

```
<?php $author = "Bill Gates";

$text = "Measuring programming progress by lines of code is like measuring aircraft building progress by weight. - $author.";

?>
```

PHP-də həmçinin <<< operatordan istifadə edərək multixətt ardıcıllıqdan istifadə etmək olar ki, bunada adətən here-document (bura sənəddir) və ya heredoc deyilir. heredoc mətndə sətirdən-sətirə keçmə və başqa boş sahələri (boşluqlar daxil olmaqla) ehtiva edən literaldır və cərgə ilə göstərişinin üsulunu təşkil edir. heredoc-dan istifadə nümunə 3.8-də göstərilmişdir.

Nümunə 3.8. *Eyni zamanda bir neçə sətirdən istifadə edən echo təlimatının daha bir variantı*

```
<?php
$author = "Brian W. Kernighan";
echo <<<_END
Debugging is twice as hard as writing the code in the first place.
Therefore, if you write the code as cleverly as possible, you are, by
definition, not smart enough to debug it. - $author.
_END;
? >
```

Bu kodun icrası zamanı PHP, iki _END teqinin arasında olan hər şeyi sətir kimi qəbul edir və bu zaman hər hansı dırnaq tipinin daxil edilməsinə ehtiyac yoxdur. Bu amil proqramçıya onu bildirir ki, məsələn, birbaşa olaraq PHP koduna HTML-kodlarının tam dəstini yazmaq və sonra da konkret olaraq, dinamik hissələri PHP dəyişənləri ilə əvəz etmək mümkündür.

Yadda saxlamaq məqbuldur ki, _END-in bağlayan teqi; yeni sətirin başlanğıcında ciddi meydana çıxmalıdır və bu sətirin tək tərkibi olmalıdır — ona hətta şərhləri (hətta tək boşluğu qoymaq olmaz) əlavə etməyə həll edilmir (icazə verilmir). Multixətt bloku bağlandıqdan sonra, sintaksis əvvəlki qaydasına düşür.

Yadda saxlayın: <<<_END.. ._END; heredoc-konstruksiyadan istifadə edərkən, siz n simvollarını əlavə etməməlisiniz və bunun yerinə yeni sətir komandası üçün, Enter klavişinə basmaq və yeni sətirdən başlamaq kifayət qədər sadədir. Tək və ya ikiqat dırnaqlarla bağlanmış başqa sətirlərdən fərqli olaraq heredoc konstruksiyasının daxilində əks sləşin () köməyilə simvolların ilkin təyinatı dəyişdirilmiş simvollardan (apostrof, cüt dırnaq və bu kimi simvollar), adi qaydada rahatlıqla istifadə edilə bilər.

Nümunə 3.9-da multixətt qiymətinin dəyişənə mənimsədilməsi sintaksisi göstərilmişdir.

Nümunə 3.9. *Multixətt qiymətinin dəyişənə mənimsədilməsi*

```
<?php
$author = "Scott Adams";
$out = <<<_END
Normal people believe that if it ain't broke, don't fix it. Engineers
believe that if it ain't broke, it doesn't have enough features yet. -
$author.
_ END;
?>
```

Bundan sonra $out dəyişəni iki teq arasında yerləşdirilmiş məzmunla doldurulacaq. Əgər mənimsədilmə və $out dəyişəninə bu qiyməti əlavə etmək lazımdırsa, onda = operatorunun yerinə .= operatordan istifadə etmək olar.

Diqqətli olun ki, _END teqi birinci hissəsində nöqtəli vergül qoymaq olmaz, çünki, bu halda nöqtəli vergül multixətt blokunu başlamadan kəsəcək və sintaktis təhlil zamanı — Parse error xətasına səbəb olacaq. Nöqtəli vergülü yalnız _END-in bağlanış teqindən sonra qoymaq lazımdır, hərçənd ki, blokun daxilində adi mətn simvolu kimi nöqtəli vergüldən istifadə etmək olar.

Yeri gəlmişkən, əbəs yerə _END teqinin izahını verməmişəm, çünki, PHP kodunda onun istifadəsi unikal, əhatə dairəsi isə genişdir. Siz artıq istənilən teqə məsələn _SECTION1 və ya _OUTPUT və s. üçün öz şəxsi mülahizələrinizi irəli sürə bilərsiz və bundan sonra bu teqləri dəyişənlərdən və ya funksiyalardan ayıra biləcəksiniz, çünki adətən teqlərin adının başlanğıcında sətir xətti nişanını qoyurlar; qeyd edək ki, öz proqramlaşdırma praktikanızda teqlərdən istifadə etməyə də bilərsiniz.

Mətnin multixətt nişanına sizin PHP kodunuzun oxunmasını sadələşdirən vasitədir çünki, mətn veb səhifədə əks olunanda və HTML formatlaşdırılma qaydaları qüvvəyə minəndə boş sahələr (amma $author dəyişəninin adı əvvəlki kimi dəyişən

qiymətlərinin yerləşdirməsi qaydalarına uyğun olaraq bu dəyişənin qiyməti ilə əvəz olunur) gizlənir. Tutaq ki, əgər brauzerə bu multixətt nümunələrinin icrası yüklənərsə, bu nümunələr bir neçə sətir şəklində əks olunmayacaq, çünki bütün brauzerlər yeni sətir simvollarına sadəcə boşluq kimi baxır. Amma əgər mənbə koduna baxmaq xüsusiyyəti olan brauzerdə HTML koda nəzər yetirsəniz görəcəksiniz ki, yeni bütün yeni sətir simvolları düzgün şəkildə yerləşib və mənbə kodunda bir neçə sətirdən ibarətdir.

Dəyişənlərin tipləri

PHP çox zəif dillərə aid edilir. Bu isə o deməkdir ki, dəyişənin elanı zamanı onun tipinin göstərilməsinə ehtiyac yoxdur, PHP həmişə dəyişənlərin tipini, bu dəyişənə giriş zamanı mühitdən asılı olaraq təyin edəcək.

Məsələn, ədəd sətirində bir neçə rəqəmdən ibarət olan ədəd yaratmaq və bu ədəddən n-ci rəqəmi çıxmaq olar. Aşağıdakı kod fraqmentində (nümunə 3.10) 12 345 və 67 890 ədədləri biri-birinə vurulur və nəticədə 838 102 050 ədədi alınır ki, sonra da alınan bu ədəd $number dəyişənində yerləşdirilir.

Nümunə 3.10. *Ədədin sətirə avtomatik dəyişikliyi*

```
<?php
$number = 12345 * 67890;
echo substr ($number, 3, 1);
?>
```

$number dəyişəninə qiymət mənimsənərkən, o ədəd tipində idi. Amma kodun ikinci sətirində bu dəyişənin qiyməti substr PHP-funksiyası tərəfindən çağırılır ki, $number dəyişənindən dördüncü mövqedə duran (unutmayın ki, PHP-də mövqeyin hesablanması sıfırdan başlanır) simvol alınsın. PHP-də bu məsələsinin icrası üçün, $number dəyişəni doqquz simvoldan ibarət olan sətirə çevrilir ki, substr funksiyası bu dəyişənə giriş edə və simvolu (indiki halda 1) ala bilsin.

Həminki qaydada zəruri olduqda sətiri ədədə çevirmək olar. Aşağıdakı nümunə 3.11-də $pi dəyişəninə 3.1415927 qiyməti sətir tipində mənimsədilmişdir ki, kodun üçüncü sətirindən sonra dairənin

sahəsinin hesablanması düsturunun bir hissəsi olmaq üçün irrasional ədədə avtomatik çevrilir və nəticədə 78,5398175 qiyməti alınır.

Nümunə 3.11. *Sətirin ədədə avtomatik dəyişikliyi*

```
<?php $pi = "3.1415927";
$radius = 5; echo $pi * ($radius * $radius);
?>
```

Bütün bunlar sizə onu bildirir ki, praktika zamanı dəyişənlərin tipləri barədə narahat olmağa ehtiyac yoxdur. Siz sadəcə, dəyişənə hər hansı bir mənaya malik olan qiymətləri mənimsətməlisiniz və PHP zəruri olduqda bu dəyişən tipləri özü dəyişdirəcək. Bundan sonra, əgər qiyməti ekranlaşdırmaq lazımdırsa, onda bunu tələb etmək (məsələn, echo təlimatının köməyilə) lazımdır.

Sabitlər

Sabitlər, dəyişənlər kimi, giriş informasiyasından ehtiva olunur, lakin sabitlərdə, adından göründüyü kimi bu informasiyanı dəyişmək olmaz. Daha dəqiq, sabitlərin qiyməti təyin olunduqdan sonra bu qiymətlər proqram vasitəsilə dəyişdirilə bilməz.

Məsələn, sabitlərdən sizin serverinizin kök kataloqunun yerinin (sizin saytınızın əsas fayllarını ehtiva edən qovluqlar) saxlanması üçün istifadə oluna bilər. Bu cür sabiti aşağıdakı qaydada müəyyən etmək olar:

```
define (" ROOT_LOCATION", "/usr/local/www/");
```

Sabitin tərkibinin oxunması üçün isə dəyişənlərdə olduğu kimi, eyni proseslə (lakin, sabitin adında dollar nişanından istifadə etmədən!) sabitə istinad etmək lazımdır:

```
$directory = ROOT_LOCATION;
```

İndi, qovluqların başqa konfiqurasiyasıyla başqa serverdə sizin PHP-kodunuzu buraxmağa lazım olan kimi, kodun yalnız bir sətirini dəyişdirmək lazım olan kimi.

Sabitlərin iki əsas xüsusiyyətini yadda saxlamaq əhəmiyyətlidir: onların adlarından əvvəl $ simvolu qoymaq lazım deyil və sabitləri yalnız

define funksiyasının köməyi ilə müəyyən etmək olar. Hamı tərəfindən qəbul edilmiş forma üzrə sabitlərin adlandırılmasında yaxşı olar ki, böyük hərf registrindən istifadə edəsiniz, xüsusilə əgər sizin kodunuzu digər biri oxuyacaqsa.

Qabaqcadan müəyyən edilmiş sabitlər

PHP-ın *qabaqcadan müəyyən edilmiş sabitləri* hazır məhsul şəklində istifadəçiyə təqdim edilir. Sizin kimi proqramlaşdırma təcrübəsi az olan şəxslər bu sabitlərdən nadir hallarda istifadə edilir. Bundan savayı, sehrli sabitlər kimi məlum olan sabitlər mövcuddur ki, lap əvvəldən sizin üçün faydalı ola bilərlər. Sehrli sabitlərin adlarının başlanğıcında və sonunda həmişə iki sətir xətti simvolu olur ki, təsadüf halında, şəxsi sabitlərin adları bu sabitlərin adları ilə eynilik təşkil etməsin. Sehrli sabitlər haqqında təfərrüatlar aşağıdakı cədvəldə göstərilmişdir.

Cədvəl 3.5. PHP-də sehrli sabitləri

Qabaqcadan müəyyən edilmiş sabit	**İzahı**
__LINE__	Cari sətirin nömrəsi
__FILE__	Fayl yerinin tam adı. __FILE__ həmişə özündə açılmış simvolik istinadlarla mütləq yolu ehtiva edir və müəyyən şərait yarandıqda (və ya əvvəlki versiyalarda) bu sehrli sabit __DIR__ sehrli sabitini (Fayl kataloqunun nisbi yolu) özündə ehtiva edə bilər.

3.5. Cədvəldə göstərilmiş anlayışlara növbəti fəsillərdə baxılacaq.
Sehrli sabit Təsvir cari sətirin Nömrəsi Tam (Dolu) yol faylın adı. Əgər include təlimatının daxilində istifadə olunursa, onda daxil edilmiş faylın adı qayıdır. PHP 4.0.2 versiyasında Əgər include təlimatının daxilində

istifadə olunursa, onda daxil edilmiş faylın kataloqu qayıdır. Eyni nəticəni dirname (__FILE__) funksiyasının tətbiqi də verir. Əgər kataloq adına bağlayan sleş yoxdursa, bu kataloq əsas kataloq olacaq. __FUNCTION__ sehrli sabiti (PHP 5.3.0 versiyasından etibarən əlavə edilmişdir) Funksiyanın adı. PHP 5-dən başlayaraq, elan edilmiş funksiyanın adını qaytarır (cədvəl hesabı ilə simvollar).
PHP 4-də qaytarılan qiymət həmişə kiçik registrin simvollarından təşkil olunur. (PHP 4.3.0-da əlavə edilmiş) Sehrli sabit Təsvir __ CLASS__ Sinfin adı. PHP 5-dən başlayaraq, elan edilən sinfin adını qaytarır (cədvəl hesabı ilə simvollar). PHP 4-də qaytarılan qiymət həmişə kiçik registrin simvollarından təşkil olunur. __METHOD__, (PHP 4.3.0 versiyasından etibarən əlavə edilmişdir) sinif metodunun Adı. Elan edilmiş metodun adını qaytarır (cədvəl hesabı ilə simvollar). __NAMESPACE__ (PHP 5.0.0 versiyasından etibarən əlavə edilmişdir) cari ad sahəsinin (cədvəl hesabı ilə simvollar) Adı. Bu sabit kompilyasiya zamanı müəyyən edilmişdir. (PHP 5.3.0 versiyasından etibarən əlavə edilmişdir) Bu sabitlər kod sətirini yerləşdirmək lazım olduqda faydalıdır: echo "Bu sətirdir ". __ LINE__. "faylda". __ FILE__; Bu komanda ilə brauzerə indi (ona yol daxil olmaqla) icra edilən cari faylın göstərişi vasitə proqramın cari sətirini ekranlaşdırılacaq.

echo və print komandalarının arasında fərq

Brauzer vasitəsilə serverdən mətn nəticəsinin ekranlaşdırılması üçün istifadə etdiyimiz echo komandasının müxtəlif üsulları artıq bizə tanışdır. Bir nümunədə sətir literalı ekranlaşdırması, digərində isə əvvəlcə sətirlərin bitişdirilməsi və ya dəyişənlərin qiymətlərinin hesablanması hadisəsinin şahidi olduq. Həmçinin bir neçə sətirə yayılan ekranlaşmanın da şahidi olduq.
Amma echo komandasına alternativ print komandası var ki, əks etmə zamanı həmçinin bu komandadan da istifadə etmək olar. Bu iki komanda bir-birinə çox oxşardır, amma print — bir tək parametr qəbul edən və qaytarılan qiymətə (hansı ki, həmişə 1-ə bərabər) malik olan funksiyaya oxşar konstruksiyadır, amma echo — təmiz PHP konstruksiyasıdır.
Ümumiyyətlə, adi mətnin ekranlaşdırılması zamanı echo komandası print-dən daha sürətli işləyir, bir çünki, echo qaytarılan qiyməti tənzimləmir. Digər tərəfdən, echo funksiya deyil və print-dən fərqli olaraq, daha mürəkkəb ifadənin bir hissəsi kimi echo-dan istifadə etmək

olmaz. Aşağıdakı nümunədə göstərildiyi kimi, dəyişənin qiymətinin həqiqi (TRUE) və ya yanlış (FALSE) olduğunu əks etdirmək üçün print funksiyasından istifadə olunur. Analoji kodu echo komandasının köməyilə ekranlaşdırmaq mümkün deyil, çünki belə olduqda kodun təhlili zamanı — Parse error sintaksis xətası barədə məlumat veriləcək:

```
$b? print "TRUE": print "FALSE";
```

Sual işarəsi, ən sual vermək üsuludur ki, $b dəyişəninin qiymətinin həqiqi və ya yanlış olduğunu biləsiniz. İki nöqtədən solda yerləşən komanda, əgər $b dəyişəni həqiqi qiymətə malikdirsə, iki nöqtədən sağda yerləşən komanda isə, əgər $b dəyişəni yanlış qiymətə malikdirsə yerinə yetirilir.
Bununla belə burada göstərilən nümunələrdə əksər hallarda echo komandasından istifadə olunur və mən sizə məhz bu komandadan istifadə etməyi məsləhət görürəm, çünki hazırlanma vaxtı sizə kodunuza print funksiyasını cəlb etmək lazım olmayacaq.

Funksiya

Funksiyalar, konkret məsələni yerinə yetirən kodun bloklara ayrılması üçün istifadə olunur. Məsələn, əgər sizə tez-tez hansısa bir məlumatları axtarmaq və müəyyən formatda bu məlumatları çıxarmaq lazım olursa, onda funksiyaya müraciət etmək ən ağıllı üsuldur. Kitabdakı funksiyalar, hələlik bir neçə sətirdən ibarət olan bloklarda göstəriləcək. Əgər zaman keçdikcə, siz məlumatların nəticəsinin formatını dəyişdirməyi istəyəcəksinizsə, funksiyaya kodun yerləşməsi onu bildirəcək ki, sizə proqramın yalnız bir yerində dəyişikliklər etmək lazım olacaq. Kodun funksiya şəklində yerləşdirməsi yalnız mənbə kodunun ölçüsünü azaltmır, həmçinin kodu oxunmasını üçün daha rahat edir, həm də koda əlavə funksional imkanlar verir. Çünki, funksiyalara parametrlər ötürülə bilər ki, bu parametrlər işin gedişatına təsir edə bilər. Funksiyalar həmçinin də kodlarda qiymətlərin alınmasına səbəb olur.
Funksiyanı nümunə 3.12-də göstərilmiş şəkildə yaratmaq olar.

Nümunə 3.12. Funksiyanın sadə elanı

```
<?php
function longdate ($timestamp)
{
return date (" l F jS Y", $timestamp);
```

```
}
?>
```

Bu funksiyaya giriş məlumatlarları UNIX sisteminin vaxt qiymətləri (tarixi əks etdirən tam ədəd və saniyə miqdarı əsasında vaxt, tarix göstərişi 1 yanvar 1970-ci ildən dəstəklənir) tərəfindən təmin olunur və sonra date PHP-funksiyası lazımlı sətir formatı (indiki halda məs., "Çərşənbə axşamı may 2 2017") ilə tarixin göstərilməsinə səbəb olur, formatında tarixi qaytarmaq üçün. Sonra yumru mötərizələrlə funksiya adına dəyənin (duranın) arasında parametrlərin istənilən miqdarını yerləşə bilər, amma bu funksiya üçün yalnız bir parametrin qəbulu seçilmişdir. Bütün kod, hansı ki, sonrakı funksiya çağırışı vaxtı (yanında) yerinə yetirilir, fiqurlu mötərizələrə nəticələnir (ibarətdir. Bu funksiyanın köməyi ilə bugünkü tarixi göstərmək üçün, öz kodunuza aşağıdakı çağırışı yerləşdirmək lazımdır: echo longdate (time()); UNIX və ötürülmə vaxtının cari qiymətinin göstərilməsi üçün bu çağırışda o longdate-ın indi yaradılmış funksiyasına, hansı ki, sonra təsvir üçün uyğun olan sətiri echo komandasına qaytarır, time-in inteqrasiya edilmiş PHP-funksiyası istifadə olunur. Əgər on yeddi günlük köhnəliyin tarixini çıxarmaq tələb olunursa, növbəti çağırışı etmək lazımdır: echo longdate (time() - 17 * 24 * 60 * 60); hansında ki, longdate funksiyaları 17 gün (17 gün × 24 ç × 60 mina × 60) ərzində keçmiş saniyələrin miqdarına azaldılmış UNIX vaxtının cari qiyməti ötürülür. Funksiyalar həmçinin bir neçə parametr qəbul edə və bir neçə nəticə qaytara bilər, texnologiyadan istifadə edərək, hansı ki, növbəti başçılarda (fəsillərdə) baxılacaq.

Dəyişənin görünmə sahəsi

Əgər proqram kodu çox uzundursa, onda dəyişənlərə ad seçimində çətinliklər yaranır. Belə hallarda, bir çox dillərdə olduğu kimi, PHP dilində də, dəyişənin görünmə sahəsini müəyyən etmək olar. Başqa sözlə, məsələn, `$temp` dəyişənini yalnız konkret funksiyanın daxilində istifadə olunacaq, funksiyandan kənar heç bir yerə tətbiq edilməyək. PHP-də faktiki olaraq məhz belə dəyişənlərin görünmə sahəsi susmaya görə təyin edilir.

Alternativ olaraq PHP-yə məlumat vermək olar ki, dəyişən qlobal görünmə sahəsinə malikdir və ona giriş proqramın istənilən yerindən həyata keçirilə bilər.

Lokal dəyişənlər

Lokal dəyişənlər funksiyanın daxilində yaradılır və onlara giriş yalnız bu funksiyanın daxilində əlçatandır. Adətən bu tip dəyişənlər funksiyadan çıxışa qədər qismən emal edilmiş nəticələri saxlamaq üçün istifadə edilən müvəqqəti dəyişənlərdir.

Lokal dəyişənlərin növlərindən biridə funksiyanın arqumentləridir. Əvvəlki bölmədə $timestamp adında parametr qəbul edən funksiya müəyyən edilmişdi. Bu parametrin qiyməti yalnız funksiyanın daxilində əlçatandır, funksiyanın xaricində isə bu dəyişəndən qiyməti almaq almaq və onu təyin etmək olmaz.

Lokal dəyişənin daha bir nümunəsini göstərmək üçün, longdate funksiyanın bir az daha dəyişdirilmiş variantına (nümunə 3.13) baxaq.

Nümunə 3.13. *Longdate funksiyasının genişləndirilmiş versiyası*

```
<?php
function longdate ($timestamp)
{
      $temp = date (" l F jS Y", $timestamp);
      return "Tarix: $temp";
}
?>
```

Bu nümunədə `date` funksiyasıyla böyüdülmüş qiymət müvəqqəti olaraq `$temp` dəyişəninə mənimsənilir ki, sonra müəyyən edilən funksiya vasitəsilə qaytarılan sətirə tətbiq edilsin. Funksiyadan çıxan kimi, `$temp` dəyişəninin qiyməti silinir, sanki belə bir dəyişəndən ümumiyyətlə heç vaxt istifadə olunmayıb.

İndi isə, dəyişənlərin görünmə sahələrinə baxmaq üçün, nümunə 3.14-də göstərilmiş oxşar kodla tanış olaq. Burada $temp dəyişəni hələ longdate funksiya çağırışına qədər yaradılmışdı.

Nümunə 3.14. *Longdate funksiyasında* `$temp` *dəyişəninə giriş əldə etməyin uğursuz cəhdi*

```
<?php
$temp = "Tarix: ";
echo longdate (time());
function longdate ($timestamp)
{
      return $temp. date (" l F jS Y", $timestamp);
}
?>
```

Uğursuzluğun səbəbi odur ki, `$temp` dəyişəni longdate funksiyasının daxilində yaradılmamış, həmçinin bu funksiyaya parametr kimi verilməmişdir, bu səbəbdən də longdate funksiyasından $temp

dəyişəninə əlçatan deyil. Buna görə də kodun bu fraqmenti yalnız bu koda qədər olan mətnsiz tarixi ekranlaşdıracaq. Əslində əvvəlcə qeyri-müəyyən dəyişəndən istifadə edilməsi barədə xəbərdarlığı bildirən (**`Notice:`** `Undefined variable: temp`) xəta göstəriləcək. Xətanın səbəbi odur ki, susmaya görə funksiyanın daxilində yaradılmış dəyişənlər yalnız bu funksiya üçün lokaldır, funksiya xaricində yaradılmış istənilən dəyişən isə, funksiya daxilində istifadə edilməyən kodda əlçatandır.

3.15 və 3.16 nümunələrində 3.14 nümunəsində göstərilən kodun düzgün formaları göstərilmişdir.

Nümunə 3.15. *Problemi* `$temp` *dəyişənini funksiya ilə birlikdə istifadə edərək həll etmək olar.*

```
<?php
$temp = "Tarix: ";
echo $temp. longdate (time());
function longdate ($timestamp)
{
      return date (" l F jS Y", $timestamp);
}
?>
```

Nümunədə `$temp`-də 3.15 istinad (sürgün) funksiyanın hüdudları üçün yerini dəyişilmişdir (köçürdülmüşdür). Bu istinad (sürgün) həminki görünmə sahəsində meydana çıxır, hansında ki, dəyişən müəyyən edilmişdi.

Nümunə 3.16*. Alternativ həll:* `$temp` *dəyişənin arqument kimi ötürülməsi*

```
<?php
$temp = "Tarix: ";
echo longdate ($temp, time());
function longdate ($text, $timestamp)
{
      return $text. date (" l F jS Y", $timestamp);
}
?>
```

Nümunə 3.16-da başqa qərar qəbul edilmişdir: longdate funksiyasına əlavə arqument kimi `$temp` dəyişəninin qiymətini vermək. Longdate funksiyası müvəqqəti $text dəyişəninin qiymətini oxuyaraq və arzuolunan nəticəni ekranlaşdırır.

```
<?php
$temp = "The date is ";
echo longdate($temp, time());
function longdate($text, $timestamp)
{
      return $text . date("l F jS Y", $timestamp);
}
```

```
?>
```
Proqramçılar dəyişənlərin görünmə sahəsini unudaraq tez-tez unudaraq səhv edir. Buna görə də, əgər dəyişənlərin görünmə sahələrinin iş prinsiplərini xatırlamasaq, aşkar edilməsi çətin xətalara səbəb olaraq. Bunu yadda saxlayın ki, əgər siz dəyişəni hər hansı xüsusi elan etməmisinizsə, bu dəyişənin görünmə sahəsi lokal sahəylə məhdudlaşır: və ya cari funksiyanın kodunun daxilində, yaxud heç bir funksiyaya aid olmayan kodun daxilində ondan asılı olaraq, harada bu dəyişən ilk dəfə yaradılmışdı və ya harada ona ilk dəfə giriş əldə edilmişdi, funksiyanın daxilində və ya onun hüdudları xaricində.

Qlobal dəyişənlər

Elə hallar olur ki, dəyişənin qlobal görünmə sahəsində elan edilməsi zərurəti yaranır ki, bu cür yaradılmış dəyişənə proqramın istənilən kodundan giriş etmək mümkündür. Bundan başqa bəzi məlumatlar o qədər həcmli və mürəkkəb ola bilər ki, onları arqument şəklində funksiyaya daxil etmək arzuolunan deyil.

Dəyişəni qlobal görünmə sahəsini elan etmək üçün, global açar sözü istifadə olunur. Fərz edək ki, saytınıza hər hansı üsul vasitəsilə istifadəçi girişi nəzərdə tutulmuşdursa və proqramın bütün kodunun sayta daxil olan şəxsin istifadəçi yaxud qonaq statusunda olduğunu müəyyən etməsi zərurəti yaranıb. Bu məsələnin həlli üsullarından biri `$is_logged_in qlobal` dəyişənin yaradılmasından ibarətdir:

```
    global $is_logged_in;
```

Yalnız sistemə girişin indi sizin funksiyasına bu dəyişən qiymət (məna) 1-i mənimsəməyə sayta girişin uğurlu cəhdi vaxtı lazımdır, və (amma) uğursuz cəhd vaxtı — qiymət (məna) 0. Bir halda ki, (çünki,) dəyişən qlobal görünmə sahəsinə malikdir, ona giriş sizin proqramınızın kodunun istənilən sətirindən alınmış ola bilər.

Amma qlobal dəyişənlərdən ehtiyatla istifadə etmək lazımdır. Mən yalnız ehtiyac yarandıqda qlobal dəyişənlərdən istifadə etməyi məsləhət görürəm. Ümumiyyətlə kiçik fraqmentlərə və ayrı məlumatlara bölünmüş proqramlar daha az xəta və daha az xidmət ehtiva edir. Əgər proqramınızın kodu bir neçə min sətirdən ibarətdirsə (bu sahə ilə məşğul olsaz, nə vaxtsa belə olacaq) və proqramın icrası zamanı məlum olur ki, qlobal dəyişən səhv qiymətə malikdir. Belə olan halda isə, kodda olan xətanın axtarışı vaxtınızı xeyli aparacaq.

Bundan başqa, əgər həddən artıq çox qlobal dəyişən yaradılıbsa, bu dəyişənlərin adlarının lokal sahədə olan dəyişələrin adları ilə eynilik təşkil etməsi riski çoxalır. Belə vəziyyətlər, müxtəlif anlaşılmayan xətalara gətirib çıxarır.
Mən bütün qlobal dəyişənlərin adlarının yazılışında böyük hərf registrindən istifadənin tərəfdaram. Belə olan halda dəyişənin görünmə sahəsini ilk baxışdan müəyyən etmək mümkün olur.

Statik dəyişənlər

"Lokal dəyişənlər" paraqrafının əvvəlində qeyd edilmişdi ki, funksiyadan çıxdıqdan sonra dəyişənin qiyməti dərhal silinir. Əgər funksiya dəfələrlə çağırılırsa, o dəyişənin yeni surəti öz işinə başlayır və onun əvvəlki parametri heç bir qiymətə malik olmur.
Maraqlıdır, əgər funksiyanın daxilində proqram kodunun digər hissələrindən girişə əlçatan olmayan lokal dəyişən varsa və bu dəyişənin qiymətinin növbəti funksiya çağrışına qədər saxlanılmasına ehtiyac yaransa, onda nə etməli? Belə hallarda statik dəyişənlərdən istifadə edilir. Nümunə 3.17-də statik dəyişənin elanı göstərilmişdir.
Nümunə 3.17. *Statik dəyişəndən istifadə edən funksiya*

```
<? php
function test()
{
      static $count = 0;
      echo $count;
      $count++;
}
?>
```

Bu nümunədə funksiyanın ən birinci sətirində `$count` adında statik dəyişən yaradılır və bu dəyişənə ilk olaraq sıfır qiyməti mənimsədilir. Növbəti sətirdə dəyişənin qiyməti ekranlaşdırılır və son sətirdə bu qiymət bir vahid böyüyür.
Funksiyanın növbəti çağırışı zamanı, `$count` dəyişəni artıq elan edildiyinə görə funksiyanın birinci sətiri buraxılır və əvvəlki funksiyanın çağırışı nəticəsində bir vahid böyüyən $count dəyişəninin qiyməti əks olunur.
Statik dəyişənlərdən istifadə etməyi planlaşdırarkən, nəzərə almaq lazımdır ki, bu tip dəyişənlərin təyini zamanı onlara hər hansı ifadə nəticəsini mənimsətmək olmaz. Bu tip dəyişənlər yalnız qabaqcadan müəyyən edilmiş qiymətlərlə (nümunə 3.18) inisializasiya oluna bilər.
Nümunə 3.18. *Statik dəyişənlərin mümkün və yolverilməz elanları*

```
<? php
      static $int = 0; // Olar
      static $int = 1+2; //Olmaz (sintaktisin səhvini doğuracaq
      (Parse error))
      static $int = sqrt (144); // Olmaz
?>
```

Superqlobal dəyişənlər

PHP 4.1.0 versiyasından başlayaraq, qabaqcadan müəyyən edilmiş bəzi dəyişənlər əlçatan oldu. Onları superqlobal dəyişənlər kimi adlandırırlar. Bu adın mənası ondan ibarətdir ki, onlar PHP mühiti mühitiylə verilirlər və proqramın daxilində qlobal görünmə sahəsinə malikdirlər, yəni istənilən onun yerindən tamamilə əlçatandır (mümkündür).

Bu superqlobal dəyişənlərdə axan işləyən proqram və onun mühiti haqqında bir çox faydalı məlumat olur (cədvəl. 3.6). Belə dəyişənlər assosiativ massiv (hansı ki, 6-cı fəsildə baxılacaq) strukturuna malikdir.

Cədvəl 3.6. *PHP-də superqlobal dəyişənlər*

Dəyişənin adı	**Tərkibi**
`$GLOBALS`	Cari ssenarinin qlobal görünmə sahəsində müəyyən edilmiş bütün dəyişənlər. Dəyişənlərin adları isə massiv açarları kimi göstərilir
`$_SERVER`	Ssenarilərin yerləşdiyi yerlər, yollar başlıqlar haqqında informasiya. Bu massivin elementləri veb-serverlə yaradılır və hər bir veb-serverin informasiyanı tam olaraq və ya onun hansısa bir hissəsini verəcəyini zəmanət vermək olmur.
`$_GET`	Cari ssenariyə HTTP GET metoduyla ötürülən dəyişənlər
`$_POST`	Cari ssenariyə HTTP POST metoduyla ötürülən dəyişənlər

$_FILES	Cari ssenariyə HTTP POST metodu vasitəsilə yüklənmiş elementlər
$_COOKIE	HTTP cookies vasitəsilə cari ssenariyə verilmiş dəyişənlər
$_SESSION	cari ssenariyə əlçatan olan sessiya dəyişənləri
$_REQUEST	İnformasiyanın tərkibi, brauzerdən verilmiş (çatdırılmış); susmaya görə $_GET,
$_ENV	Dəyişənlər, verilmişlər (çatdırılmışlar) cari ssenariyə environment metoduyla

Bütün superqlobal dəyişənlərin ($GLOBALS-dan başqa) adlarında sətir xətti olur və adlandırmada yalnız böyük hərflərdən istifadə olunur. Yaratdığınız dəyişənlərin adları superqlobal dəyişənlərin adları ilə üst-üstə düşməməlidir. Superqlobal dəyişənlərin tətbiqinə misal olaraq praktiki nümanəni göstərək. Bu nümunədə istifadəçi cari veb səhifənin URL-ünvanı alınır:

```
$came_from = $_SERVER ['HTTP_REFERRER'];
```

Gördüyünüz kimi, burada mürəkkəb heç nə yoxdur. Əgər istifadəçi bilavasitə səhifənizə daxil olduqda, məsələn brauzerdə bilavasitə URL-ünvanı daxil etsə, $came_from dəyişəninə boş sətir mənimsədiləcək.

Superqlobal dəyişənlərin təhlükəsizlik problemləri

Diqqətli olun ki, saytınıza hücum və müdaxilə vasitələri tapmağa çalışan cinayətkarlar tez-tez superqlobal dəyişənlərdən istifadə edir. Onlar $_POST-a, $_GET-ə və ya başqa superqlobal dəyişənlərə zərərverici kod - məsələn UNIX və ya MySQL komandaları yükləyirlər ki, əgər siz onlara bilmədən bu kodlara müraciət etsəz, saytınız "hack" ediləcək.

Bu səbəbdən də superqlobal dəyişənlərin tətbiqindən əvvəl həmişə onları ilkin emala məruz qoymaq lazımdır. Bunun üçün htmlentities PHP-funksiyasından istifadə etmək olar. Bu funksiya bütün HTML

simvollarının dəyişikliyi ilə məşğul olur. Məsələn, "kiçikdir" və "böyükdür" (< və >) simvolları < və > sətirlərinə çevrilir;, həminki özünü bütün dırnaqların təhlükəsiz vəziyyətinə tərcümə (köçürmə) üçün hazırlanır, əks sleşey və s.
`$_SERVER`-ə girişin buna görə daha (daha çox) yaxınlaşan üsulu (və başqa superqlobal dəyişənlərə) buna bənzəyir:
$came_from = htmlentities ($_SERVER ['HTTP_REFERRER']);
`Htmlentities` funksiyasının müayinəsi üçün istifadə yalnız superqlobal dəyişənlər barəsində, həm də istənilən şərait vaxtı (yanında) əhəmiyyətli təcrübə hesab edilmir, istifadəçidən hansı ki, məlumatlara, çıxanlara və ya kənar mənbələrdən hərəkət edənlərə, çıxış məlumatlarının alınması üçün emal edilirlər.
Bu başçıda (fəsildə) PHP-la iş üçün lazım olan etibarlı əsaslar qoyulmuşdular (girov qoyulmuşdular). 4-cü fəsildə biz ifadələrin və proqramın gedişi idarə etməsinin qurulması üçün öyrənilmiş materialın praktik istifadəsinə başlayacağıq, başqa sözlə, real proqramlaşdırmaya keçək.
Amma yeni başçının (fəsilin) öyrənilməsindən əvvəl (qarşısında) mən öz biliklərini yoxlamağı məsləhət görürəm, sonra göstərilən suallara (məsələlərə) cavab verib, ondan əmin olmaq üçün ki, siz bu başçının (fəsilin) tərkibini tamamilə mənimsədiniz.
Suallar (məsələlər)
Sual (məsələ) 3.1
PHP-ın hansı teqi əsas olur proqram kodunun interpretasiyasına başlamaq üçün? Necə bu teqin qısa forması görünür (baxır)?
Sual (məsələ) 3.2
Teqlərin hansı iki növü şərhlərin əlavə edilməsi üçün istifadə olunur?
Sual (məsələ) 3.3
Hansı simvol PHP-ın hər təlimatının sonunda durmalıdır?
Sual (məsələ) 3.4
Hansı simvol PHP-ın bütün dəyişənlərinin adlarının başlanğıcında istifadə olunur?
Sual (məsələ) 3.5
Nə dəyişənlərdə saxlanıla bilər?
Sual (məsələ) 3.6
Nədə $variable ifadələrinin arasında fərq = 1 və $variable == 1?
Sual (məsələ) 3.7
Siz hesab etdiyi kimi, niyə altından xətt çəkmələr (qeyd etmələr) dəyişənlərin (məsələn, $current_user) adlarında istifadə etməyə icazə

verilmişdir (həll edilmişdir), və (amma) defislər — yoxdur (məsələn, $current-user)?
Sual (məsələ) 3.8
Hərflərin registrinə dəyişənlərin adları həssasdırmı?
Sual (məsələ) 3.9
Dəyişənlərin adlarında boşluqlardan istifadə etmək olarmı?
Sual (məsələ) 3.10
Necə başqa tipin (deyək, saya sətirə) qiymətinə (mənasına) dəyişənin bir tipinin qiymətini (mənasını) dəyişdirmək?
Sual (məsələ) 3.11
Nədə ++$j-ın və $j++-ın arasında fərq?
Sual (məsələ) 3.12
&& və and operatorları qarşılıqlı əvəz olunandırlar?
Sual (məsələ) 3.13
Necə multixətt nəticə (çıxardılma) yaradılır: echo və ya multixətt qiymətin (mənanın) mənimsənməsiylə komandasından istifadəylə?
Sual (məsələ) 3.14
Sabiti yenidən təyin etmək olarmı?
Sual (məsələ) 3.15
Necə dırnağın ilk təyinatını dəyişdirmək?
Sual (məsələ) 3.16
Nədə echo və print komandalarının arasında fərq?
Sual (məsələ) 3.17
Funksiyaların necə təyinatı?
Sual (məsələ) 3.18
Necə dəyişəni PHP-proqramın bütün kodu üçün əlçatan (mümkün) etmək?
Sual (məsələ) 3.19
Hansı iki üsulla bütün qalan proqrama məlumatları vermək (çatdırmaq) olar, hansılar ki, funksiyanın daxilində yaradılmışdılar?
Sual (məsələ) 3.20
Nə sayla sətirin birləşməsinin nəticəsidir?
Bu suallara (məsələlərə) cavablar əlavədə (proqramda) A tapmaq olar, "3-cü fəsil məsələlərinə Cavablar" bölməsind

4

İfadələr

Əvvəlki fəsildə qeyd edilən mövzular bu fəsildə daha dolğun izah ediləcək (məsələn seçim (budaqlanma) və mürəkkəb ifadələrin yaradılması). 3-cü fəsildə mən diqqəti PHP sintaksisi və bu dillə işləməyin ən ümumi suallarında cəmləşdirmək istəyirdim, amma bu halda daha yüksək səviyyəli mövzularına toxunmamaq qeyri-mümkün idi. Amma indi sizə bütün güclü tərəfləri ilə PHP-nin dəyərli istifadəsi üçün lazım olan əsasları təqdim etmək olar.
Bu fəsildə PHP-də proqramlaşdırma təcrübəsinin təməli qoyulacaq və proqramın icrası prosesi idarə edilməsinin əsas üsullarına baxılacaq.

İfadələr

İstənilən proqramlaşdırma dilinin baza hissəsi — ifadələrdir.
İfadə qiymətlərin, dəyişənlərin, operatorların və funksiyaların uyğunluğunu təşkil olunaq, hesablanması nəticəsində yeni qiymətin yaranmasından ibarətdir. İfadələr, orta məktəb cəbrindən tanışdır, məsələn:

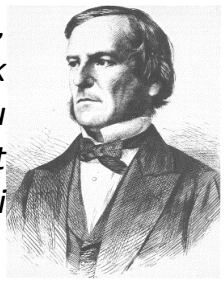
```
y = 3 (abs (2x) + 4)
```

bu ifadə PHP-də aşağıdakı formada yazılır:

```
$y = 3 * (abs (2*$x) + 4);
```

Alınan qiymət (indiki halda y və ya $y) ədəd, sətir və ya Bull (məntiqi) qiyməti (XIX əsrin ingilis riyaziyyatçısı və filosofu olan Corc Bulun şərəfinə belə adlandırılmışdır). Qiymətlərin birinci iki tipi artıq məlumatınız olmalıdır, buna görə də mən üçüncü tipi izah edəcəyəm.

Corc Bul *(**ing.** George Boole; **d.** 2 noyabr 1815, Linkoln — **ö.** 8 dekabr 1864, Ballintempl, Kork qraflığı, İrlandiya) — ingilis riyaziyyatçısı. 1849-cu ildən Kork Kral kollecinin (hal-hazırda Universitet kolleci Kork) riyaziyyat üzrə professoru. Riyazi məntiqin banilərindən biri.*



TRUE və ya FALSE?

Elementar **Bull qiyməti** ya həqiqi — `TRUE`, ya da yanlış — `FALSE` olur. Məsələn, 20 > 9 (20 9-dan böyükdür) ifadəsi həqiqi (`TRUE`), 5 == 6 (5 6-ya bərabərdir) ifadəsi isə yanlışdır (`FALSE`). (Bull və ya məntiqi əməliyyatlar zamanı həmçinin `AND`, `OR` və `XOR` operatorlarından da istifadə etmək olar və onlar haqqında məlumat bu fəsildə mövcuddur.)

Nəzərə alın ki, TRUE və FALSE adlarında üçün mən böyük hərf registrindən istifadə edirəm. Bu onu bildirir ki, onlar PHP-də qabaqcadan müəyyən edilmiş sabitdir. Bir halda ki, onlar həmçinin qabaqcadan müəyyən edilmiş sabitdir, onda istəkdən asılı olaraq həmçinin kiçik hərf registrini tətbiq etmək olar. Yeri gəlmişkən, kiçik hərf registri ilə yazılış özü-özlüyündə daha etibarlıdır, çünki bu halda PHP sabitin yenidən təyinini gözləmir, amma böyük hərf registrindən istifadə edən versiyada sabitlər yenidən təyin edilmiş ola bilər, buna görə yad kodun idxalı zamanı bunu nəzərə almaq lazımdır.

Nümunə 4.1-də bəzi sadə ifadələr göstərilmişdir: iki. Hər sətirdə a-dan d-ə qədər olan hərflərin müşayiətilə ifadələrin doğru ya yanlış olduğu ekranlaşdırılır.

Nümunə 4.1. *Dörd sadə Bull ifadəsi*

```
<?php
  echo "a: [". (20 > 9). "] <br>";
  echo "b: [". (5 == 6). "] <br>";
  echo "c: [". (1 == 0). "] <br>";
  echo "d: [". (1 == 1). "] <br>";
?>
```

Bu kod növbəti informasiyanı çıxaracaq:

```
a: [1]
b: []
c: []
d: [1]
```

Nəzərə alın ki, `a:` və `d:` ifadəsinin hesablanmasının nəticəsi, 1 qiymətini ehtiva edən həqiqi (`TRUE`) qiymətdir. Amma `b` və `c` ifadələrinin hesablanmasının nəticəsi yanlışdır (`FALSE`) və belə olan halda heç bir qiymət göstərilmir, bir çünki, PHP-da `FALSE` sabiti `NULL` (heç nə) kimi müəyyən edilmişdir. Bundan əmin olmaq üçün, nümunə 4.2-də göstərilən kodu yoxlamaq olar.

Nümunə 4.2. `TRUE` *və* `FALSE` *qiymətlərinin nəticəsi*

```
<?php
 //test2.php
 echo "a: [". TRUE. "] <br>";
 echo "b: [". FALSE. "] <br>";
?>
```

Bu kod aşağıdakı informasiyanı ekranlaşdıracaq:

```
a: [1]
b: []
```

Yeri gəlmişkən, bəzi dillərdə `FALSE` sabiti `0` və ya hətta `-1` kimi müəyyən edilə bilər, buna görə də istifadə etdiyiniz dildə bu sabiti yoxlamağa dəyər.

Literallar və dəyişənlər

İfadə həmçinin sadə dəyişən də ibarət ola bilər ki, bu zaman dəyişənə mənimsənmiş qiymət hesablanır. Hər iki forma ifadələrin tiplərinə aiddir, çünki, onlar qiyməti (mənanı) qaytarırlar. Nümunə 4.3-də üç literal və iki dəyişən göstərilmişdir və göstərilənlərin hər biri müxtəlif tiplərdə qiymət alır.

Nümunə 4.3*. Literal və dəyişənlər*

```
<?php
  $myname = "Brian";
  $myage = 37;

  echo "a: ". 73. "<br>"; // Ədəd literalı
  echo "b: ". "Hello". "<br>"; // Sətir literalı
  echo "c: ". FALSE. "<br>"; // Sabit literalı
  echo "d: ". $myname. "<br>"; // Sətir dəyişəni
  echo "e: ". $myage. "<br>"; // Ədəd dəyişəni
? >
```

Çıxış informasiyasında siz `c` ifadəsindən başqa bütün ifadələrdə gözlənilən qiyməti alacaqsınız, çünki hesablanmanın nəticəsi `FALSE`-dir və bu halda heç bir qiymət alınmır:

```
a: 73
b: Hello
c:
```

```
d: Brian
e: 37
```

Operatorlar

Operatorlarla ən sadə ifadələri birləşdirib, daha mürəkkəb ifadələri yaratmaq olar ki, bu da hər hansı hesablanma əməliyyatını daha faydalı edəcək.

Mənimsənmə və ya idarə edici konstruksiya ifadələr ilə birləşərkən təlimat alınır. Nümunə 4.4-də hər növdən bir təlimat göstərilmişdir. Nümunələrdən birincisində ifadələrə `366` nəticəsinin mənimsənməsi həyata keçirilir - `$days_to_new_year`, `$day_number` dəyişəni, və (amma) ikincidə təbrik xəbəri (mesajı) çıxarılır, əgər `$days_to_new_year` ifadəsi `< 30 TRUE` kimi hesablanırsa.

Nümunə 4.4. *İfadə və təlimat*

```
<?php
  $days_to_new_year = 366 - $day_number; // Expression
  if ($days_to_new_year < 30)
  {
    echo "Not long now till new year";  // Statement
  }
?>
```

PHP-də mənimsəmə, müqayisə və bir çox başqa operatorlardan tutmuş, hesab, sətir və məntiqi operatorlara qədər bir çox güclü operatorlar mövcuddur.

Müxtəlif tiplərdə olan operatorlar müxtəlif miqdarda operand qəbul edirlər.

- Artım `($a++)` və ya ədəd işarəsi dəyişmə `(-$a)` kimi unar operatorları, yalnız bir operandı qəbul edirlər.
- PHP, böyük miqdarda operator dəstini təqdim edir. Toplama, çıxma, vurma və bölmə operatorları daxil olmaqla ikili operatorlar, iki operand qəbul edir.
- x formasına malik olan bir üçkomponentli operator `? y: z`. Faktiki olaraq, bu yazılış, birsətirli üç hissədən ibarət olan if təlimatıdır ki, üçüncü ifadənin hesablanmasının nəticəsindən asılı olaraq iki ifadənin arasında seçim həyata keçirir.

Operatorların prioritetliyi

Əgər bütün operatorlarda prioritetlik eyni səviyyədə olsaydı, onda operatorlar interpretatorun gördüyü düzülüşdə emal ediləcəkdi. Nümunə 4.5-də göstərildiyi kimi faktiki olaraq bir çox operatorların prioritetliyi eyni səviyyədədir.

Nümunə 4.5. *Üç ekvivalent ifadə*

```
1 + 2 + 3 – 4 + 5
2 + 1 - 4 + 5 + 3
5 + 1+ 2 - 4 + 1 + 3
```

Nümunədən göründüyü ki, rəqəmlərin (və onlardan əvvəl olan operatorlarun) yer dəyişdirməsinə baxmayaraq, hər bir ifadənin nəticəsi 7 qiymətidir, çünki, "üstəgəl" və "çıxma" operatorlarının prioritetliyi eyni səviyyəlidir. Analoji qayda ifadəni vurma və bölmə operatorları ilə tətbiq etdikdə də keçərlidir.

Nümunə 4.6. *Üç ekvivalent ifadə*

```
1 * 2 * 3 / 4 * 5
2 / 4 * 5 * 3 * 1
5 * 2 / 4 * 1 * 3
```

bu nümunədə alınan qiymət həmişə 7,5-ə bərabərdir. Amma, əgər ifadədə prioritet səviyyəsi müxtəlif olan operatorlar mövcuddursa, bu zaman operatorların icrası prioritet səviyyəsinə müvafiq olaraq həyata keçirilir (nümunə 4.7).

Nümunə 4.7. *Prioritet səviyyəsi müxtəlif olan operatorlardan təşkil olunmuş üç ifadə*

```
1 + 2 * 3 - 4 * 5
2 - 4 * 5 * 3 + 1
5 + 2 - 4 + 1 * 3
```

Əgər operatorların prioritetliyi mövcud olmasaydı, onda bu ifadələrin hesablanması nəticəsində müvafiq olaraq 25, −29 və 12 ədədi alınardı. Amma, vurma və bölmə operatorları toplama və çıxma operatorları ilə müqayisədə daha yüksək prioritet səviyyəsinə malikdir.

İfadələrdə operatorların prioritetlik səviyyəsini idarə etmək üçün mötərizələrdən istifadə etmək olar və əgər yuxarıda göstərilən ifadədə bunu tətbiq etsək, nümunə 4.8-də göstərilən formada ifadə alınacaq.

Nümunə 4.8. *Mötərizələr ilə idarə olunan üç ifadələr*

```
1 + (2 * 3) - (4 * 5)
2 - (4 * 5 * 3) + 1
5 + 2 - 4 + (1 * 3
```

Göründüyü kimi, PHP əvvəlcə mötərizələr ilə bağlanmış alt ifadəni hesablamalıdır ki, nümunə 4.9-da göstərilən qismən hesablanmış ifadələr alınsın.

Nümunə 4.9. *Mötərizələr ilə idarə olunan üç ifadələr*

```
1 + (6) - (20)
2 - (60) + 1
5 + 2 - 4 + (3)
```

Bu ifadələrin hesablanmasının son nəticəsi müvafiq olaraq −13 −57 və 6 (bu da operatorların prioritetliyinin yoxluğunda alınan 25 −29 və 12 nəticələrindən tamamilə fərqlənir) bərabərdir.

Operatorların ardıcıllığı

Biz operatorların prioritetliyi qüvvəyə minmədiyi halda ifadələrin soldan sağa emal olunduğunu görərik. Amma bəzi operatorlar sağdan sola emalı tələb edə bilər.

Emalın istiqaməti operatorların ardıcıllığına səbəb olur. Ayrı-ayrı operatorlar üçün ardıcıllıq yoxdur. Siz açıq-aydın prioritetlikləri dəyişdirən zaman ardıcıllıq böyük əhəmiyyət kəsb edir. Bunun üçün sizə operatorların susmaya görə hərəkətləri haqqında bilmək lazımdır.

Nümunə 4.11-də hər üç dəyişənə 0 qiyməti mənimsənilir.

Nümunə 4.11. Çoxsaylı mənimsəmə operatoru

```
<?php
$level = $score = $time = 0;
?>
```

Belə çoxsaylı mənimsəmə yalnız o halda mümkündür ki, ifadənin əvvəlcə ən sağ tərəfi hesablansın və sonra da proses sağdan sola davam etsin.

PHP ilə yeni işləyərkən məqbuldur ki, operatorların ardıcıllığı məsələsində (haradakı hesablamada səhv ola

bilər) daha soyuqqanlı davranaraq, yumru mötərizələrdən istifadə edərək, hesablama prosesini öz istəyinizə uyğun tənzimləyin. Bu sizə və başqa proqramçılara, kodun daha rahat başa düşülməsinə səbəb olacaq.

Müqayisə operatorları

Müqayisə operatorları iki operandın qiymətlərini yoxlayırlar və bərabər, TRUE, və ya FALSE məntiqi qiymətlərini alır. Müqayisə operatorlarının üç tipi mövcuddur: bərabərlik, müqayisə və məntiqi operatorlar.

Bərabərlik operatoru

Biz bu kitabda ilk dəfə deyil ki, bərabərlik operatorlarına (iki bərabərdir simvolu ilə) rast gəlirik. Bu operatoru = (tək bərabərdir simvolu) mənimsəmə operatoru ilə qarışdırmaq olmaz .

Nümunə 4.12-də birinci operator qiymət mənimsəyir, ikinci operator isə bu qiymətin bərabərlini yoxlayır.

Nümunə 4.12. Qiymətin mənimsədilməsi və qiymətin yoxlanılması

```
<?php
  $month = "Mart";
  if ($month == "Mart") echo "Yaz gəldi";
?>
```

Nümunədən göründüyü kimi, TRUE və ya FALSE qiymətlərini alan müqayisə operatoru if təlimatından istifadə edərək şəraiti yoxlamağa imkan verir. Amma bu hələ hər şey deyil çünki, PHP zəif tipləşdirilmiş dildir. Əgər bərabərlik olan ifadənin hər iki tərəfi müxtəlif tiplərə malik operandlardan ibarətdirsə, PHP bu operandları ən böyük mənaya malik operandın tipinə dəyişdirəcək.

Məsələn, rəqəmlərdən tamamilə təşkil edilmiş istənilən sətir, ədədlərlə müqayisə zamanı ədəd tipinə dəyişdiriləcək. Nümunə 4.13-də $a və $b dəyişəni iki müxtəlif sətirdir və buna görə də if təlimatlarından hansısa bir nəticənin çıxacağını gözləmək çətindir.

Nümunə 4.13. Bərabərlik və eyniliyin operatorları

```
<?php
  $a = "1000";
  $b = "+1000";
  if ($a == $b) echo "1";
  if ($a === $b) echo "2";
?>
```

Amma əgər bu nümunəni icra etsək, görərik ki, həqiqətən də rəqəmlər ekranlaşdırılır. Bu onu bildirir ki, birinci if təlimatının hesablanmasının

nəticəsi TRUE qiymətidir. Onda belə çıxır ki, hər iki sətir əvvəlcə ədədlərə konversiya edilir və 1000 ilə +1000 eyni ədəd qiyməti olduğu bilinir.

Birincidən fərqli olaraq, ikinci if təlimatında eynilik operatorundan (üç qat bərabərlik işarəsi) istifadə olunur. Buna görə də $a və $b dəyişənləri sətir kimi müqayisə edilir və indi bir-birindən fərqli hesab edildiyinə görə, heç nə ekranlaşdırılmır.

Necə ki, operatorların prioritetlik səviyyəsinin idarə etmək üçün mötərizələrdən istifadə edilirdi, eləcə də, əgər PHP operandların tiplərini konversiya etməsi barədə şübhə varsa və ya interpretatorun belə davranışının ləğvi üçün eynilik operatorun istifadə etmək olar. Operandların bərabərliyinin təyini üçün bərabərlik operatorunun tətbiq edildiyi kimi, analoji olaraq bərabərsizliyin təyini üçün bərabərsizlik (!=) operatorundan istifadə edilir. Nümunə 4.14, 4.13 nümunəsinin dəyişdirilmiş formasıdır ki, bərabərlik və eynilik operatorlarının əksi olan operatorlar ilə əvəz edilmişdir.

Nümunə 4.14. Bərabərsizlik və eynilik deyil operatorları

```
<?php
$a = "1000";
$b = "+1000";
if ($a! = $b) echo "1";
if ($a! == $b) echo "2";
?>
```

Gözlənildiyi kimi, birinci if təlimatı 1 rəqəmini ekranlaşdırır, çünki kodda $a və $b dəyişənlərinin ədəd qiymətlərinin bərabərsizliyi haqqında sual verilir.

Bunun yerinə, ikinci if təlimatı 2 rəqəmini ekranlaşdıracaq, çünki kodda $a və $b dəyişənlərinin operandlarının əvvəlki tipinin (sətir tipinin) eynilik olmaması haqqında sual verilir və TRUE cavabı alınır, çünki həqiqətən də bu operandlar eyni deyil.

Müqayisə operatorları

Müqayisə operatorları istifadə edərək, operandların yalnız bərabərliyini və bərabərsizlikliyini deyil, geniş yoxlanış əhatəsi malik bir sıra əməliyyatları həyata keçirmək olar.

Bunun üçün PHP sizə > (böyük), < (kiçik), >= (böyük və ya bərabərdi) və <= (kiçik və ya bərabərdir) operatorlarından istifadə etməyə imkan verir.

Nümunə 4.15-də bu operatorlardan istifadə göstərilmişdir.

Nümunə 4.15. Dörd fərqli müqayisə operatoru

```
<?php
```

```
$a = 2;
$b = 3;
if ($a > $b) echo "$a $b-dən böyükdür<br>";
if ($a < $b) echo "$a $b-dən kiçikdir<br>";
if ($a >= $b) echo "$a $b-dən böyük və ya bərabərdir<br>";
if ($a <= $b) echo "$a $b-dən kiçikdir və ya bərabərdir<br> ";
?>
```

Bu nümunədə, $a dəyişəni 2 qiyməti, $b dəyişəninə isə 3 qiyməti mənimsədilib, ekrana aşağıdakı informasiya çıxır:

```
2  3-dən kiçikdir və ya bərabərdir
```

Bu nümunədə $a və $b dəyişənlərinin qiymətlərini dəyişdirərək, nümunə ilə müstəqil davranmağa cəhd edin.

Məntiqi operatorlar

Məntiqi operatorlar həqiqi və ya yanlış nəticələr alır. Dörd məntiqi operator mövcuddur.

Bu operatorların istifadəsi nümunə 4.16-da göstərilmişdir. Nəzərə alın ki, PHP NOT sözünün yerinə ! simvolundan istifadə etməyi tələb edir. Bundan başqa, operatorlar kiçik hərf və ya böyük hərf registrindən təşkil edilə bilər.

Nümunə 4.16. Məntiqi operatorlardan istifadə

```
<?php
$a = 1;
$b = 0;
echo ($a AND $b). "<br>";
echo ($a or $b). "<br>";
echo ($a XOR $b). "<br>";
echo! $a. "<br>";
?>
```

Bu nümunə NULL, 1, 1, NULL nəticəsi ekranlaşdırılır. Bu isə o deməkdir ki, yalnız ikinci və üçüncü echo təlimatları hesablama nəticəsində TRUE qiymətini alır. (Xatırlamaq lazımdır ki, NULL və ya heç nə, FALSE qiymətini əks etdirir.) Çünki AND operatoruna, TRUE qiymətini qaytarmaq üçün, hər iki operand həqiqi qiymətə malik olmalıdır

Proqramlaşdırmayla məşğul olduqda, xatırlamaq lazımdır ki, AND və OR operatorları && və || operatorlarına nisbətən daha aşağı prioritet səviyyəsinə malikdir. Buna

görə də mürəkkəb ifadələrdə && və || operatorlarının tətbiqi yəqin ki, daha təhlükəsizdir. OR operatorunun if təlimatında istifadəsi gözlənilməyən problemlərə səbəb ola bilər, çünki əgər birinci operandın hesablanması nəticəsində artıq TRUE qiyməti alınmışsa ikinci operand hesablanmayacaq.

Nümunə 4.17-də əgər $finished dəyişəni 1 qiymətini ehtiva edirsə getnext funksiyası heç vaxt çağırılmayacaq.

Nümunə 4.17. OR operatorundan istifadə edən təlimat

```
<?php
if ($finished == 1 OR getnext() == 1) exit;
?>
```

Əgər getnext funksiyasının hər bir if təlimatına ücün çağırılmasını istəyirsinizsə, nümunə 4.18-də göstərilmiş kodu daxil edin.

Nümunə 4.18. If təlimatında dəyişikliklər... Getnext funksiya çağırışını zəmanət verən OR

```
<? php
$gn = getnext();
if ($finished == 1 OR $gn == 1) exit;
?>
```

Bu halda getnext funksiyada olan kod yerinə yetiriləcək və qaytarılmış qiymət $gn dəyişənində hələ if təlimatının icrasına qədər saxlanacaq.

Başqa qərar (həll) onunla nəticələnir (ibarətdir), yerlərlə şəraitin (şərtlərin) sadə yer dəyişdirməsinin hesabına getnext funksiyasının icra edilməsini təmin etmək üçün, bir halda ki, (çünki,) onda funksiya çağırışı ifadədə birincilərə meydana çıxacaq.

Cədvəldə. 4.5 məntiqi operatorlardan istifadənin bütün mümkün variantları göstərilmişdir. Qeyd etmək lazımdır ki,! TRUE FALSE ekvivalentidir, və (amma)! FALSE — TRUE ekvivalenti.

Şərtlər

Şərtlər proqramın icrasının prosesini dəyişdirir. Şərt vasitəsilə konkret məsələlərə uyğun həll yolu seçmək olar. Şərtlər dinamik veb səhifələrin hazırlaması zamanı mühüm rol oynayır — PHP-dan istifadənin əsas məqsədləri, bir halda ki, (çünki,) informasiyanın veb səhifəsinə hər baxış vaxtı çıxarılanın müxtəlif variantlarının yaradılmasını yüngülləşdirirlər.

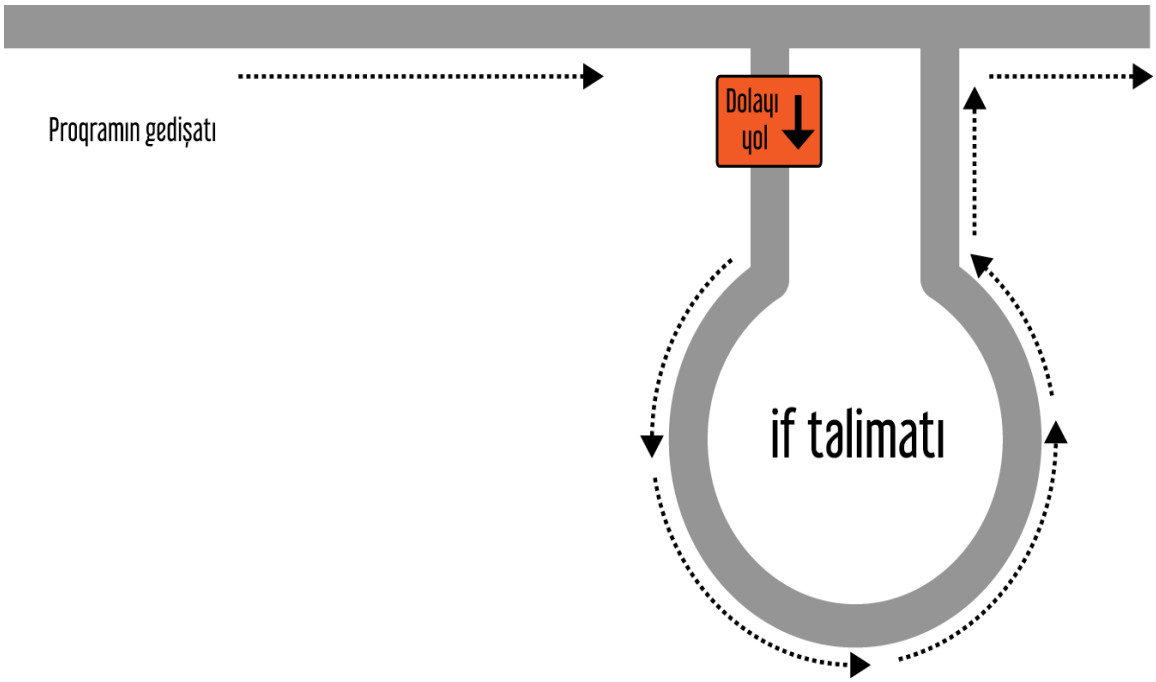
Qeyri-dövri şərti təlimatların üç tipi mövcuddur: if, switch və?. Onları ona görə qeyri-dövri adlandırıram ki, bu təlimatlarla yerinə yetirilən

əməliyyatlardan sonra proqramın icrası prosesi davam edir, amma dövri şərti təlimatlardan (hansı ki, hələ tanış deyilik) istifadə vaxtı isə müəyyən şərtə riayət edilənə qədər kod yenidən yerinə yetirilir.

if təlimatı

Proqramın icrası prosesini birzolaqlı magistral yolda hərəkət edən maşın kimi təsəvvür edin. Bu magistral yol çox hissəsi düzdür, amma bəzən hərəkət istiqamətini göstərən müxtəlif yol nişanlarına da rast gəlinir.

if təlimatı ilə qarşılaşdıqda, təsəvvür etmək olar ki, maşın dolayı yol nişanına yaxınlaşır. Bu zaman da təlimata uyğun şərtlər daxilində TRUE qiyməti hesablanır. Bu halda siz magistral yoldan çıxırsız və dolayı yolla hərəkət edirsiniz (bu halda da sizin ilk marşrut xəttiniz dəyişir). Və ya, əgər şərt TRUE kimi hesablanmırsa, siz (şəkil dolay yola məhəl qoymursunuz və heç bir şey olmamış kimi magistrallar üzrə getməyə davam edirsiniz. 4.1).



Şəkil. 4.1. Proqramın icrasının (ifasının) prosesi magistralın odnopolosnoy-u üzrə (görə) hərəkətə oxşardır

If şərt təlimatının tərkibinə istənilən PHP-ifadəsi daxil edilə bilər (bərabərlik, müqayisə, sıfır və NULL yoxlanılması və hətta funksiyalarla (müstəqil hazırlanmış, həm də yaradılmış) qaytarılan qiymətlər daxil olmaqla).

Şərtin TRUE hesablanması zamanı edilən əməliyyatlar bir qayda olaraq, fiqurlu mötərizələrin {} daxilində yerləşir. Amma əgər cəmi bir təlimat yerinə yetiriləcəksə fiqurlu mötərizələrdən istifadə etməmək olar. Bununla belə, hər bir halda fiqurlu mötərizələrdən istifadə etsək yaxşıdır, çünki şərtə əlavə əməliyyatlar daxil edildikdə fiqurlu mötərizələr unudularsa "xəta ovçusuna şikar olacağıq" (Nəzərə alın ki, yerin iqtisadiyyatı və materialın anlaşılması üçün, əgər kitabda gətirilən nümunələrdə cəmi bir icra edilən təlimat idi, mən bu məsləhəti (şuranı) izləmirdim (getmirdim) və fiqurlu mötərizələri salırdı.)
Nümunə 4.19-a müvafiq olaraq təsəvvür edin ki, ayın sonu yaxınlaşıb və bütün hesablar üzrə olan ödənişləri həyata keçirtmək lazımdır. Buna görə də siz bank hesabıyla bəzi əməliyyatları keçirirsiniz.
Nümunə 4.19. If təlimatı, hansı ki, fiqurlu mötərizələrdə istifadə olunur

```
<? php
if ($bank_balance < 100)
{
$money = 1000;
$bank_balance += $money;
}
?>
```

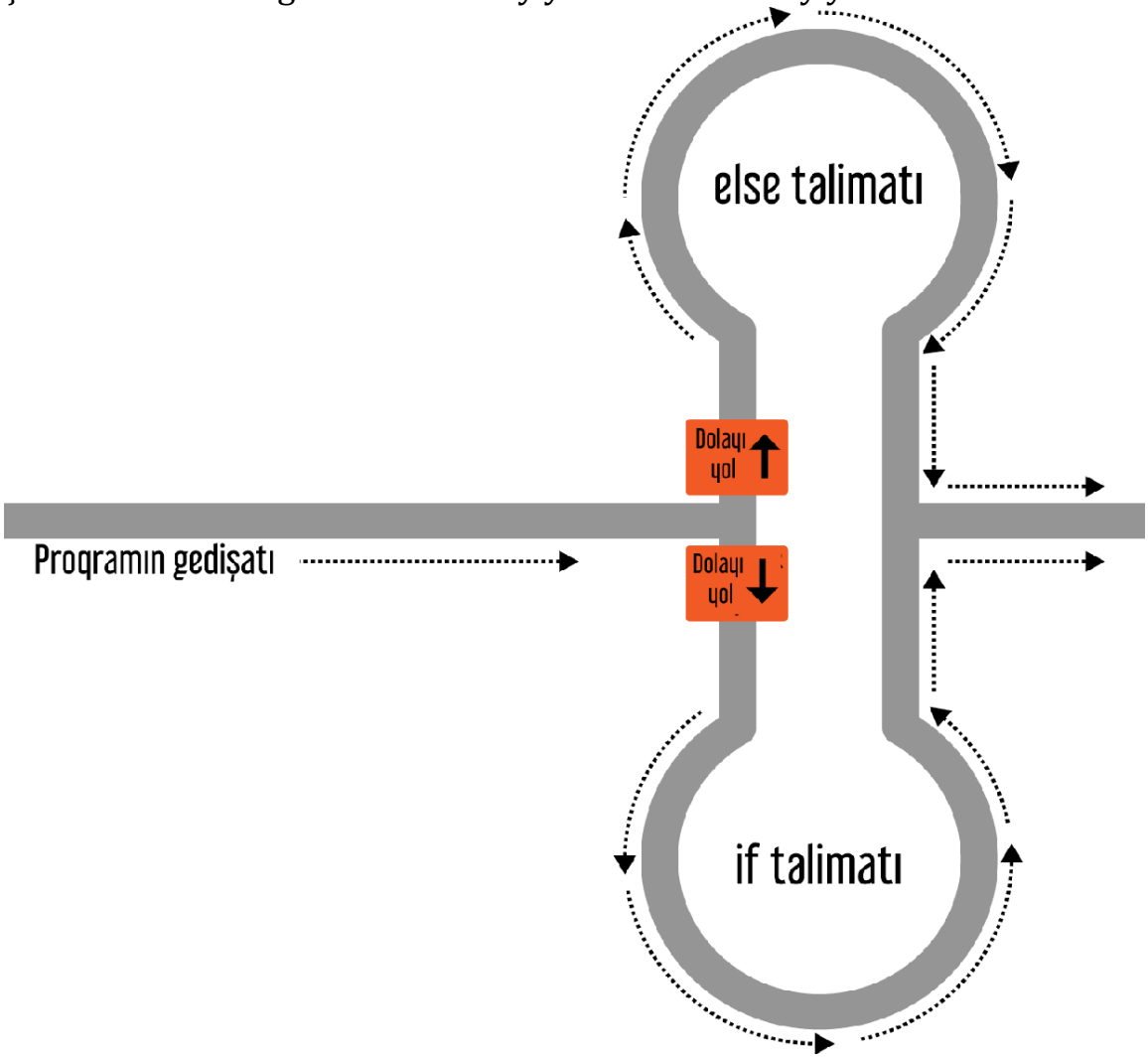
Bu nümunədə balansın $100 aşağı olması yoxlanılır. Əgər balans bu məbləğdən aşağıdısa, siz $1000 ödəyirsiniz və sonra ödədiyiniz pulu balansa əlavə edirsiniz. (Yaxşı pulu qazanmağa belə sadədir!)
Əgər bankdakı hesabın balansı $100-a bərabərdirsə və ya məbləği ötürsə, şərtdəki təlimatlara məhəl qoyulmur və proqramın icrası prosesi kodun (hansı ki, burada göstərilməmiş) növbəti sətirinə keçir.
Bəzi proqramçılar birinci fiqurlu mötərizəni şərt ifadəsindən sağda, bəziləri isə yeni sətirdə qoymağa üstünlük verirlər. Bu kitabda fiqurlu mötərizələrin açılışı adətən yeni sətirdə yerləşir. Burada heç bir fərq yoxdur, çünki PHP mülahizənizə uyğun hər hansı boş sahələrin (boşluqlar, yeni sətirlər və tabulyasiya simvolları) yerləşdirilməsinə icazə verir. Amma kodun daha oxunaqlı və qaydaya salınması daha asan olması üçün, şərtlərin hər birə səviyyəsini tabulyasiya simvolunun köməyi ilə formalaşdıra bilərsiniz.

else təlimatı

Elə hallar olur ki, şərt TRUE kimi hesablanmır, amma proqramın əsas kodunun icrasına dərhal davam etməsini istəmirsiniz və bunun yerinə başqa bir əməliyyatı etmək istəyirsiniz. Bu halda else təlimatı işinizə

yarayacaq. Əvvəlki metaforaya qayıtsaq magistral yolunuzda else-nin köməyiylə ikinci dolayı yolu təşkil etmək olar. 4.2.

Şəkil. 4.2. İndi magistralda if dolay yolu və else dolay yolu var



Əgər if...else konstruksiyasından istifadə zamanı şərt TRUE kimi hesablanırsa, onda birinci şərt təlimatı yerinə yetirilir. Amma əgər bu şərt FALSE kimi hesablanırsa, onda ikinci şərt təlimatı yerinə yetirilir. Bu iki təlimatdan biri icra üçün seçilmiş olmalıdır, amma hər iki birdən (dərhal) onlar nə hansı şərait (şərtlər) vaxtı (yanında) yerinə yetirilməyəcək və mütləq heç olmasa onlardan biri yerinə yetiriləcək. If konstruksiyasından istifadə.. .else nümunə 4.20-də göstərilmişdir.

Nümunə 4.20. If konstruksiyası.. .else, hansında ki, fiqurlu mötərizələr istifadə olunur

```
<? php
if ($bank_balance < 100)
{
```

$money = 1000;
$bank_balance += $money;
}
else
{
$savings += 50;
$bank_balance - = 50;
}
?>

Əgər bu nümunədə bankda $100 və ya bu məbləğdən daha çox olarsa, onda else təlimatı yerinə yetirilir ki, bu halda pulun bir hissəsi sizin əmanət hesabınıza yerləşdirilir.

if-də olduğu kimi, əgər else təlimatında yalnız bir şərt təlimatı varsa, onda fiqurlu mötərizələri qoymamaq olar. (Hərçənd ki, fiqurlu mötərizələrdən istifadə hər bir halda tövsiyə edilir. Birincisi, fiqurlu mötərizələrin mövcudluğu zamanı kodu anlamaq daha sadə olur, ikincisi isə, bu mötərizələr təlimatları budaqlandıraraq, təlimatların sonrakı əlavəsini yüngülləşdirir.)

`elseif` *təlimatı*

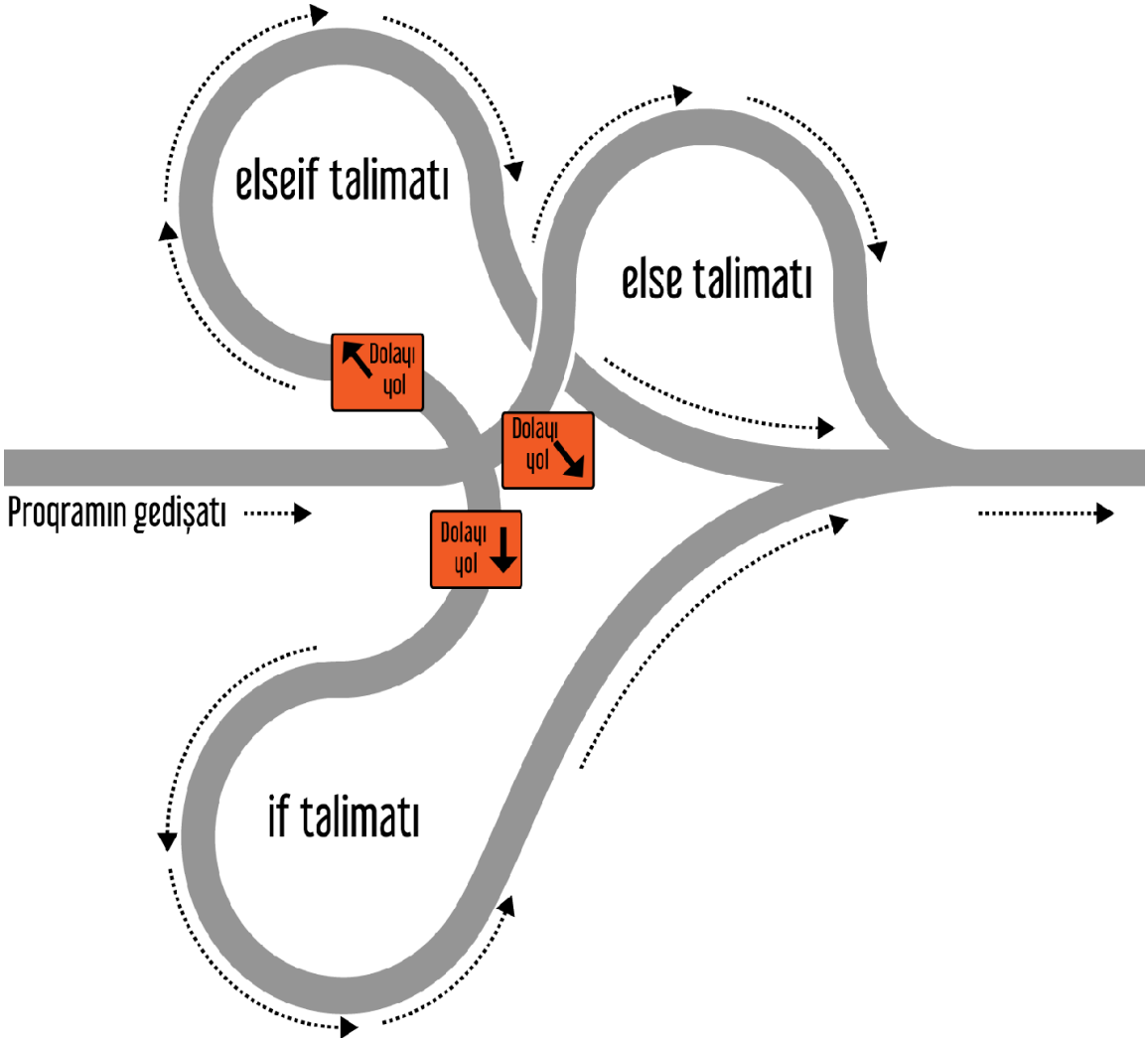
Tutaq ki, şərt ardıcıllığı əsasında eyni anda bir neçə əməliyyatı həyata keçirmək tələb olunur. Bu halda, elseif təlimatından istifadə edərək arzu olunan nəticəyə nail olmaq olar. Fərz etmək olar ki, bu təlimat else təlimatına oxşardır, lakin bu təlimatda şərt koduna qədər daha bir şərt ifadəsi qoyulur. If-in dəyərli konstruksiyası.. .elseif.. .else nümunə 4.21-də göstərilmişdir.

Nümunə 4.21. If konstruksiyası.. .elseif.. .else, hansında ki, fiqurlu mötərizələr istifadə olunur

```
<?php
 if ($bank_balance < 100)
 {
   $money = 1000;
   $bank_balance += $money;
 }
 elseif ($bank_balance > 200)
 {
   $savings += 100;
   $bank_balance - = 100;
 }
 else
 {
```

```
    $savings += 50;
    $bank_balance - = 50;
  }
?>
```

bu nümunədə elseif təlimatı `if` və `else` təlimatlarının arasında yerləşdirilmişdir. Bu nümunədə elseif təlimatı, bank hesab balansındakı məbləğin 200$-ı ötdüyü yoxlayır və əgər göstərilən məbləğ 200$-dan çoxdursa, onda bu ay əmanət hesabına 100$ daxil edir. Bütün bunları bir neçə istiqamətdə uzanan dolayı yol şəklində göstərmək olar (şəkil. 4.3).



Şəkil. 4.3. If, elseif və else dolayı yolları ilə magistral Else təlimatı qurtarır və ya if konstruksiyasını.. .else, və ya if konstruksiyasını.. .elseif.. .else. Əgər labüd deyilsə, else təlimatını axıra saxlamağınız məsləhətdir, amma heç bir halda istər elseif, istərsə də else təlimatı if təlimatından

öncə qurula bilməz. elseif təlimatları qeyri-məhdud istifadə malikdir. Amma bu təlimatların miqdarı çoxdursa, switch təlimatına müraciət etmək daha yaxşı və məntiqli üsuldur.

`switch` *təlimatı*

Tutaq ki, bir dəyişəndə və ya ifadənin hesablanmasının nəticəsində bir neçə qiymət ola bilər ki, bu qiymətlərin hər biri üçün xüsusi funksiyanın çağrılması lazımdır. Belə hallarda switch təlimatından istifadə edilir. Məsələn, aşağıdakı PHP kodu ilə idarə olunan menyu sisteminə baxaq. Bu kodun icrası ilə, istifadəçinin arzusuna uyğun olaraq ayrı-ayrı sətiri əsas menyunun koduna daxil edilir. Fərz edək, aşağıdakı variantlar mövcuddur:

Home, About, News, Login və Links, — və istifadəçi tərəfindən daxil edilmiş informasiyaya uyğun olaraq bu qiymətlərdən birini qəbul edən $page dəyişəni. Bu fikrin, if konstruksiyasından istifadə edərək alqoritmləşdirilməsi, nümunə 4.22-də göstərilmişdir. Nümunə 4.22. Multixəttli if təlimatı.. .elseif <? php if ($page == "Home") echo "Siz Home-ni seçdiniz"; elseif ($page == "About") echo "Siz About-u seçdiniz"; elseif ($page == "News") echo "Siz News-u seçdiniz"; elseif ($page == "Login") echo "Siz Login-i seçdiniz"; elseif ($page == "Links") echo "Siz Links-i seçdi";? > switch təlimatından istifadə olunan kod, nümunə 4.23-də göstərilmişdir. Nümunə 4.23. Switch təlimatı <? php switch ($page){ case "Home": echo "Siz Home-u seçdi"; break; case "About": echo "Siz About-u seçdi"; break; case "News": echo "Siz News-i seçdi"; break; case "Login": echo "Siz Login-i seçdi"; break; case "Links": echo "Siz Links-i seçdi"; break;}? > Gördüyünüz kimi, $page dəyişənindən yalnız bir dəfə — switch təlimatının başlanğıcında istifadə olunur. Bundan sonra bütün uyğunluqlar case komandasından yoxlanır. Uyğunluq tapılan zaman, qoyulan şərt təlimatı yerinə yetirilir. Əlbəttə ki, cari proqramda bu yerdə istifadəçiyə nəyi seçdiyi barədə sadə informasiya deyil, səhifəyə təsvirin və ya keçidin kodu tətbiq ediləcək. switch təlimatlarında olan case komandalarının daxilində fiqurlu mötərizələr istifadə olunmur. Bunun əvəzinə təlimatlar iki nöqtə ilə başlayır, break komandası vasitəsilə bitir. Bununla belə, switch təlimatında olan bütün case komandalarının siyahısı fiqurlu mötərizələrdən ibarətdir.

Switch təlimatının işinin dayandırılması

Əgər, switch təlimatı şərtin icrasına görə öz işini dayandırması lazımdırsa, break komandasından istifadə etmək lazımdır. PHP-də sözügedən komanda, switch təlimatının işini bitirir və növbəti təlimatın

icrasına keçidi təmin edir. Əgər nümunə 4.23-də break və Home şərtini yoxlayan case komandasının hesablanma nəticəsini yerbəyer etməsək, TRUE qiyməti alınacaq, onda sıra ilə case komandalarının beş şərti təlimatı yerinə yetiriləcək.
Əgər $page dəyişəni News qiymətini ehtiva edirdisə, onda bu yerdən başlayaraq, bütün case komandaları yerinə yetiriləcək. Bu qəsdən proqramlaşdırma imkanlarının genişlənməsi üçün edilmiş addımdır, amma məsləhətdir ki, case komanda sırası ilə şərti təlimatların işinin bitirilməsi lazım olan yerlərdə break komandasından istifadə edin. Qeyd etmək lazımdır ki, break komandalarından təsadüfi istifadə olduqca sıx rast gəlinən xətalardan biridir. Hərəkət (təsir) susmaya görə hərəkətə (təsirə) keçid susmaya görə switch təlimatı üçün Tipik tələbdir, əgər heç bir case komandalarında olan şəraitdən (şərtlərdən) yerinə yetirilməyəcəksə. Məsələn, nümunə 4.23-də göstərilmiş menyunun kodunda fiqurlu mötərizə ilə əhatələnmiş, bilavasitə qabaq nümunə 4.24-də göstərilmiş kodu əlavə etmək olar. Nümunə 4.24. Əlavə edilmə üçün default təlimatı məsələn 4.23 default: "Müəyyən edilməmiş seçim" echo-u; break; Hərçənd ki, burada break komandasını qoymaq tələb olunmur, çünki, default sonuncu daxili təlimatdır və proqramın icrası prosesi bitirən fiqurlu mötərizədən sonra avtomatik davam edəcək, amma əgər siz yuxarı default təlimatını bu yer qoymağa qərar verəcəksinizsə, break komandası ona şübhəsiz lazım olacaq, ki, proqramın icrasının (ifasının) prosesi aşağı bütün dəyən (duran) şərti təlimatlara toxunmasın. Yenidən sığortalanmaq və bu təlimatın sonunda həmişə break komandasını qoymaq daha yaxşıdır. Alternativ sintaksis switch təlimatının Açan fiqurlu mötərizəsini iki nöqtəylə əvəz etmək olar, və (amma) bağlayanı — endswitch (4.25 öldü) komandasıyla. Belə variant nadir hallarda kifayət qədər istifadə olunur, burda o o hadisəyə xatırlanır, əgər başqalarına kimləsə yaradılmış kodda onunla rastlaşmaq lazım olacaqsa. Nümunə 4.25. Switch təlimatının alternativ sintaksisi <? php switch ($page): case "Home": echo "Siz Home-u seçdi"; break; // və t. d. ... case "Links": echo "Siz Links-i seçdi"; break; endswitch;? >

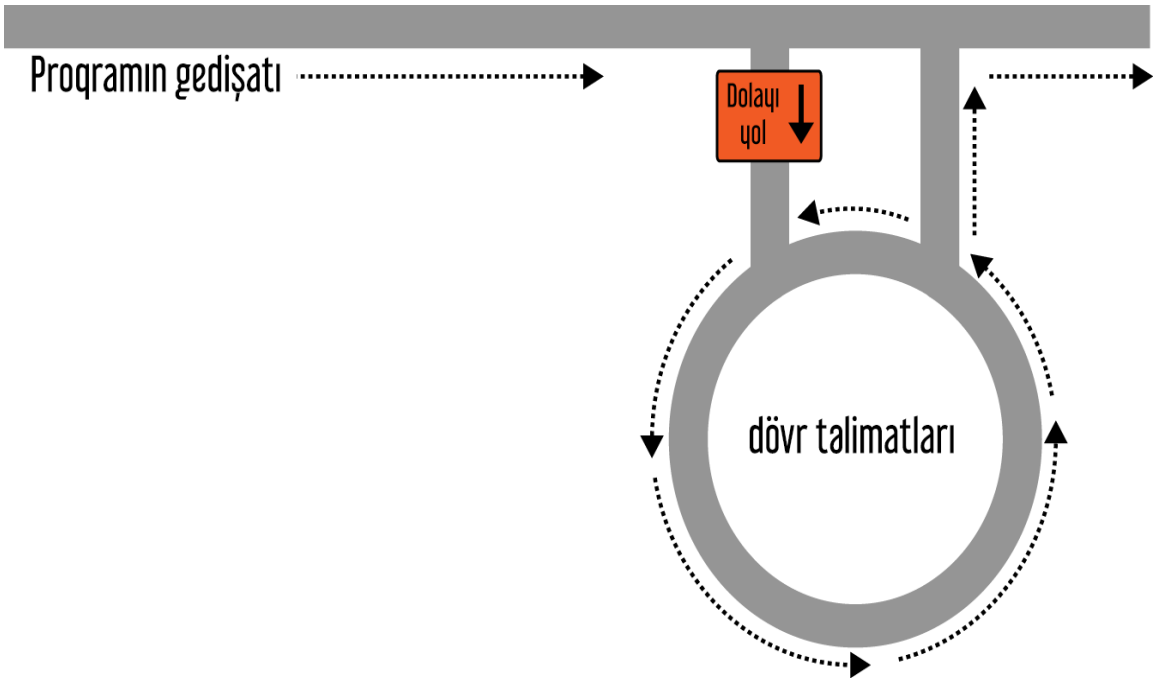
? operatoru

Üçkomponentli ? operatoru if və else təlimatlarının uzun-uzadı sintaksisini qısaldılmış variantda təqdim etməyə imkan verir. Bu operatorun qeyri-adiliyi ondan ibarətdir ki, digər əksər operatorlar kimi iki yox və üç operanddan istifadə edir. Bu operator 3-cü fəsildə print ilə

echo arasında fərqin müəyyənləşdirilməsi zamanı bizə tanışdır, hansı ki, bu müqayisədə qeyd edilirdiki, ? operatoru print ilə işləyir. ? operatoru ifadəni ötürülürlər, hansı ki, o hesablamalıdır, və iki yerinə yetirilən operator: bir icra (ifa) üçün, nə vaxt ki, TRUE ifadəsinin hesablanmasının nəticəsi, və (amma) başqa — nə vaxt ki, FALSE. Nümunə 4.26-da kod göstərilmişdir, hansı ki, cihazların onun panelinə avtomobildə yanacağın səviyyəsi haqqında xəbərdarlığın nəticəsi (çıxardılması) üçün istifadə oluna bilər. Nümunə 4.26. Operatordan istifadə? <? php echo $fuel <= 1? "Əlavə yanacaq doldurma tələb olunur": "Yanacaqlar hələ (daha) kifayət";? > Əgər yanacaqlar 1 qallon1 hamı (hər şey) qalırsa və ya daha az (başqa sözlə, $fuel dəyişəni məna kəsb edir, bərabər vahidə və ya daha az o), onda bu operator echo-un əvvəlki komandasına sətiri qaytarır "əlavə yanacaq doldurma Tələb olunur". Əks təqdirdə o "Yanacaqlar hələ (daha) kifayət" sətiri qaytarır. Operator tərəfindən qaytarılan qiymət (məna)?, həmçinin hər hansı dəyişən (nümunə 4.27) mənimsəmək olar. Nümunə 4.27. Operatorun şərti nəticəsinin mənimsənməsi "?" dəyişən <? php $enough = $fuel <= 1? FALSE: TRUE;? > 1 1 qallon (amerika) = 3,79 l. — Primeç. redaksiya. $enough dəyişəninin bu nümunəsində TRUE qiyməti (mənası) yalnız o hadisədə mənimsənəcək, əgər çəndə 1 qallondan çox yanacaq, əks təqdirdə o FALSE qiyməti (mənası) mənimsənəcək. Əgər siz operatorun sintaksisini hesab edirsinizsə? çox dolaşdırılmışlara, o onun yerinə if təlimatından istifadə edə bilərsiniz, amma onun haqqında onsuz da bilmək lazımdır, bir halda ki, (çünki,) o başqa proqramçı tərəfindən yaradılmış proqram kodunda görüşə bilər. Kodun oxunması, hansında ki, bu operator istifadə olunur, eyni dəyişənin bir neçə (bir qədər) yerində sıx tətbiqə görə güclü çətinliyə salınmış ola bilər. Məsələn, belə növün kodu olduqca məşhurdur: $saved = $saved >= $new? $saved: $new; Anlamaq ki, o edir, yalnız dəqiq təhlildən (götürmədən) sonra olar: $saved = // $saved $saved dəyişəninin qiymətinin (mənasının) Mənimsənməsi >= $new // $saved və $new Müqayisəsi? // Əgər müqayisə həqiqi nəticəni verirsə (ələ verirsə)... $saved //... $saved-in cari qiyməti (mənası) ona mənimsənilir: // Əgər müqayisə yanlış nəticəni verirsə (ələ verirsə)... $new; //... $new dəyişəninin qiyməti (mənası) ona Bu böyük əhəmiyyətin özünün izlənilməsinin olduqca yığcam üsulunu mənimsənilir, hansı ki, proqramın icrası (ifası) prosesində görüşə bilər. Özünü böyük əhəmiyyət $saved dəyişənində olur və yeni qiymətin (mənanın) daxil olması vaxtı $new dəyişəninin qiymətiylə (mənasıyla) müqayisə edilir.

Dövrlərin təşkili

Kompüterlərin əsas qabiliyyətlərindən biri təkrar hesablanma proseslərini icra etməsidir. Dövrlər, təkrarlanma prosesləri proqramlarda sıx rast gəlinən məsələlərdən biridir. PHP-də olan müxtəlif strukturlu dövrlərin təşkili belə məsələlərin çox gözəl üsullar ilə həllini təklif edir. Dövrlərin iş prinsipini təsəvvür etmək üçün şəkilə baxın.



4.4. Şəkil, if təlimatının iş prinsipinin illüstrasiyası üçün istifadə olunan metaforaya oxşayır. Lakin burada dolayı yolda bağlı sahədə var ki, avtomobil bu yoldan yalnız müəyyən proqram şəraitinə riayət etməklə çıxa bilər.

`while` *dövrləri*

Gəlin while dövründən istifadə edərək, avtomobildə hərəkət zamanı yanacağın səviyyəsinə daim yoxlayan nümunəvi kod ilə tanış olaq. Şəkil. 4.4. Proqramının dövr hissələrinin magistral yol kimi təsəvvürü Nümunə 4.28. While dövrü <? php $fuel = 10; while ($fuel > 1){ // səfərin Davamı... echo "Yanacaq hələ kifayət qədərdir";}? > Nəzərə alın

ki, əgər siz bu nümunəni icra etsəniz, onda yuxarıdakı sətir brauzer bağlanılana qədər daim ekranlaşdırılacaq. if təlimatlarında olduğu kimi, while dövrünün daxilində təlimatların saxlanılması üçün fiqurlu mötərizələrdən istifadə olunur, əgər yalnız bu dövrə yalnız bir təlimat cəlb edilməmişdir. Nümunə 4.29-da while dövründən istifadənin daha bir variantı göstərilmişdir. Bu nümunədə 12-nin vurma cədvəli göstərilir. Nümunə 4.29. 12-nin vurma cədvəlinin göstərilməsi üçün istifadə edilən while dövrü <? php $count = 1; while ($count <= 12){ echo "12 vurulsun $count bərabərdir ". $count * 12. "
"; ++ $count;}? > bu nümunədə $count dəyişəninə ilkin olaraq 1 qiyməti mənimsədilir, sonra isə $count dəyişəninin <= 12 şərtini ödəyən while dövrünə keçid edilir. Dövr, dəyişənin qiyməti 12-dən çox olana qədər yerinə yetiriləcək. Bu kod aşağıdakı mətni çıxaracaq: 12-də vurulmuş (artırılmış) say 1 12-də vurulmuş (artırılmış) 12 Say 2 bərabərdir 12-də vurulmuş (artırılmış) 24 Say 3 bərabərdir 36 bərabərdir və t. d. Sətirin nəticəsi və həmçinin 12-ə vurulmuş dəyişənin qiymətləri dövrün daxilində həyata keçirilir. Nəticənin vizual görünüşünü yeni sətir -
 teqindən istifadə edilmişdir. Sonra da PHP, dövrün sonunda qoyulmuş $count dəyişənin artımı əməliyyatını icra edib, dövrün başlanğıcına qayıdır və dövr daxilində olan kod yenidən icra edilir. Belə olan halda, $count dəyişəninin qiyməti qoyduğumuz =<12 şərtinin tələblərinə cavab verdiyini bilmək üçün yenidən yoxlanılır. Aydın məsələdir ki, dövrün birinci mərhələsində artımdan sonra $count dəyişənin qiyməti 2 olacaq və bu qiymət qoyduğumuz şərtin tələblərinə cavab verir və bu andan etibarən sonrakı 11 dövr mərhələsi keçdikdən sonra $count dəyişənin qiyməti 13-ə bərabər olacaq. Belə olan halda, qoyduğumuz şərtin tələbləri pozulacaq və bu da proqramın tamamlanması ilə nəticələnəcək. ++$count operatorunun (ekvivalent olaraq, $count++ operatoru da tətbiq edilə bilər) yoxluğunda bu dövr əməliyyatı bölmənin əvvəlində göstərilmiş nümunəyə bənzəyəcək ki,birinciyə bənzəyəcək, yəni dövr əməliyyatı heç vaxt bitməyəcək, 1 × 12 hasili təkrar-təkrar hesablanacaq və aydındır ki, eyni nəticə ekranlaşdırılacaq. Amma bu dövr yazılışının daha da zərif üsulu mövcuddur. Nümunə 4.30-dakı koda baxın. Nümunə 4.30. Nümunə 4.29-un qısaldılmış versiyası <? php $count = 0; while (++$count <= 12) echo "12-də vurulmuş (artırılmış) $count Sayı bərabərdir ". $count * 12. "
";? > bu nümunədə ++$count operatoru while dövrünün gövdəsindən silinmişdir və bilavasitə dövrün şərt hissəsinə ifadə şəklində yerləşdirilmişdir. İndi PHP dövrün (təkrarlar) hər keçidinin başlanğıcında $count dəyişəninin qiymətini hesablayır.

Dəyişənin adından əvvəl artım operatoru qoyulduğuna görə, əvvəlcə dəyişənin qiyməti bir vahid artırılır və yalnız bundan sonra artımdan alınan ədəd 12 ədədi ilə müqayisə edilir. Beləliklə, indiki halda $count dəyişəninə ilkin olaraq, 1 qiyməti deyil 0 qiyməti mənimsədilir, çünki, bu tip sintaksisdə, qiymət dövrə giriş anında dərhal artırılır. Əgər ilkin qiyməti 1 olaraq təyin etsək, onda dövr hesab əməliyyatlarını 2 ilə 12 ədədləri arasında edəcək.

`do...while` *dövrləri*

do Dövrü.. .while o hadisədə istifadə edilən while dövrünün kiçik modifikasiyasını təşkil edir nə vaxt lazımdır ki, kodun bloku heç olmasa bir dəfə yerinə yetirilmişdi, və (amma) şərt yalnız yoxlanırdı bundan sonra.
Nümunə 4.31-də bu dövrdən istifadə edərək, 12-nin vurma cədvəlinin dəyişilmiş versiyası göstərilmişdir. Nümunə 4.31. Do dövrü.. .while, 12-nin vurma cədvəlinin nəticəsi üçün istifadə edilən <? php $count = 1; do echo "12-də vurulmuş (artırılmış) $count Sayı bərabərdir ". $count * 12. "
"; while (++$count <= 12);? > Qeyd edək ki, indi biz $count dəyişəninə ilkin olaraq 1 mənimsədirik, çünki kod dəyişənin 1 qiymətinin artımı olmadan, dərhal yerinə yetirilir. Kodun qalan hissəsi nümunə 4.29-da göstərilmiş koda çox oxşardır. Əlbəttə ki, əgər do...while dövrünün daxilində bir neçə təlimat varsa, bu təlimatların ətrafında fiqurlu mötərizələri qoymağı unutmayın (nümunə 4.32). Nümunə 4.32. Nümunə 4.31-in fiqurlu mötərizələrdən istifadə edilən genişləndirilmiş versiyası <? php $count = 1; do{ echo "12-də vurulmuş (artırılmış) $count Sayı bərabərdir ". $count * 12; echo "
";} while (++$count <= 12);? >

for dövrləri

dövrün təlimatlarının son növü olan for Dövrü bundan başqa həmçinin onlardan ən güclü, bir halda ki, (çünki,) onda dövrə giriş vaxtı (yanında) dəyişənlərin qiymətinin (mənasının) quraşdırılmasının imkanları birləşir, dövrün (təkrarlar) hər keçidi və hər təkrardan sonra dəyişənlərin qiymətlərinin (mənalarının) modifikasiyası vaxtı şərtə riayət etmənin yoxlamaları. Nümunə 4.33-də for dövründən istifadə etməklə vurma cədvəlinin göstərilməsi qeyd olunmuşdur. Nümunə 4.33.

For dövründən istifadə etməklə 12-nin vurma cədvəlinin hesablanması <? php for ($count = 1; $count <= 12; ++$count) echo "12-də vurulmuş (artırılmış) $count Sayı bərabərdir ". $count * 12. "
";? > Gördüyünüz kimi, bütün kod bir for bir təlimatının daxilindədir. Buradan belə nəticəyə gəlmək olar. Hər bir for təlimatı üç parametr qəbul edir: • inisializasiyanın ifadəsi; • şərtin ifadəsi; • modifikasiyanın ifadəsi.
Bu ifadələr bir-birindən nöqtəli vergül ilə ayrılır. İlk olaraq, təkrarlanmanın başlanğıcında inisializasiyanın ifadəsi yerinə yetirilir. Bu zaman bizim vurma cədvəli kodunda $count dəyişəninə 1 qiyməti inisializasiya olunur. Sonra hər təkrarlanma zamanı şərt ifadəsi (indiki halda bu $count <= 12) yoxlanır və dövrdən çıxış yalnız o halda baş verir ki, şərtin hesablanmasının nəticəsi TRUE olur. Və nəhayət, hər təkrarlanmanın tamamında modifikasiya ifadəsi yerinə yetirilir. Vurma cədvəli kodunda bu $count dəyişəninin qiymətinin bir vahid artmasıdır. Bu strukturda, təlimatların dövri olaraq icrası mümkündür və bütün növ təlimatlar sərbəst şəkildə dövrün gövdəsinə əlavə edilə bilər. Əgər for dövrünün gövdəsində bir neçə təlimat yerləşdirmisinizsə, bu zaman fiqurlu mötərizələrdən istifadə etməyi unutmayın (nümunə 4.34). Nümunə 4.34. Nümunə 4.33-dəki for dövrünə fiqurlu mötərizələrin əlavə edilməsi <? php for ($count = 1; $count <= 12; ++$count){ echo "12-də vurulmuş (artırılmış) $count Sayı bərabərdir ". $count * 12; echo "
";}? > şərtin Cravnim-i, hansılar ki, for dövrlərindən istifadə etmək lazımdır, şəraitlə (şərtlərlə), hansılar ki, while dövrlərindən istifadə etmək lazımdır. For dövrü açıq-aydın daimi ölçü ilə dəyişənin ayrı-ayrı qiymətlərinin əsasında yaradılır. Adətən biz artan qiymət əməliyyatlarında bu dövrdən istifadə edirik. Məsələn, istifadəçiyə elementlərdən ibarət siyahı təqdim edirsiz. İstifadəçi də təqdim olunmuş siyahıdan hər hansı elementi seçir və sizdən onun seçdiyi elementi növbə ilə emal etməyiniz tələb olunur. For təlimatının daha mürəkkəbliyi ondan ibarətdir ki, bu təlimat hətta eyni zamanı hər üç parametrlə bir neçə əməliyyat həyata keçirməyə imkan verir: for ($i = 1, $j = 1; $i + $j < 10; $i++, $j++){ //... } Amma bu dili yeni öyrənənlərə belə mürəkkəb formadan istifadə etmək tövsiyə edilmir. Burada əsas vergülləri nöqtəli vergüldən ayırmaqdır. Hər üç parametr bir-birindən nöqtəli vergüllə ayırılmalıdır. Hər parametrin daxilində olan bir neçə operator bir-birindən vergüllərlə ayırılmalıdır. Əvvəlki nümunədə, birinci və üçüncü parametrlər iki operator ehtiva edir: $i = 1, $j = 1 // $i və $j $i dəyişənlərinin İnisializasiyası + $j < 10 // $i++ dövrünün işinin sonu Şərti, $j++ // Əsas hər təkrarın sonunda $i və $j Modifikasiyası, nə

bundan nümunə aydınlaşdırmaq lazımdır, — parametrlərin üç seksiyası nöqtəli vergüllə bölünməlidir, və (amma) deyil vergüllərlə (hansılar ki, yalnız parametrlərin hər seksiyasının daxilində operatorların bölməsi üçün istifadə oluna bilərlər). Onda hansı şəraitdə for təlimatlarından qabaq while təlimatlarına üstünlük vermək lazımdır? Sizin şərtiniz daimi əsasda dəyişənin sadə dəyişikliyindən asılı deyilsə. Məsələn, while təlimatı o hadisələrə tətbiq ediləcək ki, hansısa bir müəyyən qiymət daxil edilməmiş və ya hansısa bir konkret xəta olmayan və xəta yaranarkən dərhal dövrü bitirilməsi ilə nəticələnən hadisələrin yoxlanmasından ibarətdir.

Dövrün işinin dayandırılması

for dövrünün işini bitirmək üçün dəqiq və artıq switch təlimatında da qeyd edilmiş — break komandasından istifadə edilir. Məsələn, sizin təlimatlarınızdan biri xətalıdırsa və bu halda da dövrün icrasını davam etmək təhlükəli olduğu üçun dövrün işinin tamamlanması labüddür. Belə hadisələrdən biri faylın yazılışı zamanı diskdə yerin çatışmamazlığına görə yarana biləcək xətadır. Nümunə 4.35. for dövründən istifadə edən fayl yazılışı nümunəsi <? php $fp = fopen ("text.txt", 'wb'); for ($j = 0; $j < 100; ++$j){ $written = fwrite ($fp, "data"); if ($written == FALSE) break;} fclose ($fp);? > Bu kodun fraqmentləri, əvvəllər göstərilmiş kodlara nisbətən daha mürəkkəbdir, amma siz artıq bu kod anlamağa hazırsınız. Faylların emalı komandalarına növbəti fəsillərin birində baxılacaq, amma indi yalnız bilmək lazımdır ki, kodun birinci sətirində ikiqat rejimdə yazılış üçün text.txt faylı açılır və sonra da $fp dəyişəninə göstərici (siyahı) qoyulur, hansı ki, gələcəkdə bu açıq fayla istinad üçün istifadə olunacaq. Sonra 100 dövr keçidi ərzində (0-dan 99-a qədər) fayla data sətirinin yazılışı həyata keçirilir. Hər yazıdan sonra fwrite funksiyası, $written dəyişəninə müvəffəqiyyətlə yazılmış simvolların miqdarından ibarət qiyməti mənimsədir. Amma əgər səhv yaranarsa, onda fwrite funksiyası bu dəyişənə FALSE qiymətini mənimsədir. Fwrite funksiyasının belə davranışı, $written dəyişəninin yoxlanmasını yüngülləşdirir, əgər dəyişən bu qiymətə malikdirsə, kod dövrün işini dayandırır və faylı bağlanması təlimatının icrası baş verir. İstəkdən asılı olaraq bu kodu yaxşılaşdırmaq və sətiri sadələşdirmək olar: if ($w
if ($written == FALSE) break; və ya NOT operatorundan istifadə etməklə: if (! $written) break; Faktiki olaraq dövrün daxilində olan iki

təlimat bir təlimata qədər azaldılmış ola bilər: if (! fwrite ($fp, "data")) break; Amma break komandası fərz ediləsi mümkün olan daha geniş imkanlara malikdir, çünki, əgər kodun hər hansı səviyyəsinin işini qurtarmaq lazımdırsa, break komandasından sonra səviyyəni göstərən ədədi daxil etmək lazımdır, məsələn: break 2;

Continue təlimatı

continue Təlimatı break komandasına bir az oxşardır, yalnız bu təlimat PHP-də cari dövr prosesini dayandırmaqla yanaşı, bilavasitə bu prosesin növbəti təkrarına təkan verir, yəni PHP-də bütün dövr işinin dayandırılması və çıxışının yerinə, yalnız cari dövrün təkrarını həyata keçirir. Bu təlimat o hadisələrdə lazım ola bilər ki, tələ olunan artıq qiymət məlumdur və cari dövrün icrasını davam etməyə ehtiyac yoxdur. Bu təlimatdan istifadə etməklə dövr prosesində yarana biləcək arzuolunmaz xətalardan qaçmaq mümkündür. Nümunə 4.36-da sıfıra bölmə əməliyyatı zamanı yaranan arifmetik xətadan qaçmaq üçün continue təlimatından istifadə olunur. Nümunə 4.36. Continue təlimatının köməyi ilə sıfıra bölmə xətasından sığortalanma <? php $j = 10; while ($j > −10){ $j--; if ($j == 0) continue; echo (10 / $j). "
";} > $j dəyişəninin sıfırdan başqa, [10 - -10) aralığında olan bütün qiymətləri göstərilən bölmə əməliyyatında iştirak edə bilər. Amma, əgər $j dəyişənin qiyməti konkret olaraq 0-a bərabərdirsə, continue təlimatı çağırılır və sonrakı təkrarlanma prosesi dayandırılaraq, dərhal dövrün növbəti təkrarına keçirilir.

Tiplərin naməlum və açıq dəyişikliyi

PHP zəif tipləşdirilmiş dildir. Yəni, bu dildə d sadəcə dəyişəni elan etmək kifayətdir. Həmçinin zəruri olduqda bir tipin digər tipə avtomatik dəyişikliyi həyata keçirilir. Bu proses tiplərin naməlum dəyişikliyi adlanır. Ancaq elə vəziyyətlər də olur ki, PHP-yə məxsus tiplərin naməlum dəyişikliyi zamanı tamamilə arzu edilməyən nəticələr alınır. Nümunə 4.37-ə baxın və nəzərə alın ki, bölmə əməliyyatı üçün giriş məlumatlarları tam ədəddir. PHP susmaya dəqiq nəticə almaq üçün çıxış məlumatlarını kəsr ədədləri şəklində göstərir — 4,66 və 6 dövrdə. Nümunə 4.37. Bu nümunədə kəsr ədədi alınır <? php $a = 56; $b = 12; $c = $a / $b; echo $c;? > Bəs, $c dəyişəninin qiymətini tam ədəd şəklində

göstərmək üçün nə etməli? Bunu müxtəlif üsullarla etmək mümkündür ki, bu üsullardan biri (int) operatorundan istifadə etməklə dəyişənin tipinin tam ədədə çevrilməsidir. Bu zaman $a/$b əməliyyatının nəticəsinin məcburi dəyişikliyi həyata keçirilir: $c = (int) ($a / $b); Belə üsul tiplərin açıq dəyişikliyi adlanır. Nəzərə alın ki, bütün ifadənin qiymətinin tam ədəd tipinə dəyişikliyini təmin etmək üçün bu ifadə yumru mötərizələrlə əhatələnməlidir. Əks təqdirdə yalnız $a dəyişəninin qiyməti dəyişikliyə məruz qalacaq ki, bu da aydın məsələdir ki, heç bir mənaya malik olmayacaq, çünki, $b dəyişəninin qiyməti ilə bölmə əməliyyatı prosesində onsuz da kəsr ədəd alınacaq. Cədvəl də göstərilmiş tiplərin qiymətlərinin açıq dəyişikliyini həyata keçirmək olar. 4.6, amma adətən o PHP-ın inteqrasiya edilmiş funksiyalarından birinin çağırışının hesabına dəyişiklikdən istifadə edərək qaçmaq olar. Məsələn, tam ədəddən ibarət qiymətin alınması üçün intval funksiyasından istifadə etmək olar. Bu bölmə, kitabın başqa bölmələri kimi, əsasən rastınıza çıxan hər hansı yad kodu başa düşməyə kömək edir.

PHP-də dinamik əlaqə

PHP proqramlaşdırma dilidir və bu dilin işi nəticəsində alınan çıxış informasiyası müxtəlif istifadəçilər üçün müxtəlif ola bilər. PHP-nin köməyi ilə yaradılmış bir veb səhifəni tam sayt şəklində istehsal etmək olar. İstifadəçi hər hansı təsadüf halında və ya bilərəkdən ziyarət etdiyi veb səhifəyə təkrar geri qayıda bilər ki, bu zaman da cookie-in müxtəlif obyektlərinə uyğun olaraq və/və ya saxlanmış sessiya məlumatları vasitəsilə qərar qəbul edilə bilər. Bu yolla tam sayt yaradılması mümkünlüyünə baxmayaraq, bu cür üsul tövsiyə edilmir, çünki, bu halda mənbə kodu genişlənəcək və bir müddət keçdikdən sonra nəhəng ölçülü kod alınacaq. Saytı bir neçə müxtəlif hissədən ibarət ssenarilə təşkil etmək ən məntiqli üsuldur. Məsələn, saytda qeydiyyat zamanı bir avtonom proses, daxil edilən istifadəçi adının, parolun, elektron poçt ünvanlarının doğruluğunun yoxlanması ilə məşğul olacaq. İkinci modul isə, yaxşı olar ki, istifadəçilər saytınızın əsas səhifəsinə daxil olarkən, onların qeydiyyatını yaradan modul olsun. Sonra da xəbərlərin nümayişi modulunu yaratmaq olar ki, istifadəçilər öz şərhlərini və faydalı məlumatlarını daxil edə bilsin, şəkilin saytına yükləməyə icazə verən daha bir modulu və t-ı qoya (tərk edə) bilərdilər. d. Cookie və ya dəyişən sessiyalar (hər iki bu vasitə daha ətraflı növbəti başçılarda

(fəsillərdə) baxılacaq) obyektlərindən istifadə edən sizin saytınızda istifadəçinin hərəkətlərinin (təsirlərinin) izlənilməsi üçün vasitə yaradılan kimi PHP-kodun rahat seksiyalarına saytı bölmək olar, hansılardan ki, hər biri başqalardan müstəqil olacaq. Beləliklə, siz özünüzə xas, yeni xüsusiyyət hazırlaya bilərsiniz. Dinamik bağlanma hərəkətdə ol (qüvvədə qal) Ən məşhur proqramlardan biri, bloqların təşkili və idarə ediltməsini təmin WordPress platformasıdır (şəkil. 4.5). Bloq ilə tanışlığı olanlar yəqin bilirlər ki, bloqun hər bir əsas seksiyasının öz əsas PHP-faylı mövcuddur və birgə istifadə edilən çoxlu funksiyalar müxtəlif fayllarda saxlanmışdır ki, bu fayllar tələb olunan PHP sintaksisi vasitəsilə bir-birilə əlaqələndilir. Şəkil. 4.5. Bloqların təşkili üçün nəzərdə tutulmuş WordPress platforması PHP-də yazılmışdır Bütün platforma səhnə arxası sessiya əsasında işləyir, buna görə də, çətin ki, tabe edilmiş başqa bir seksiyadan keçidi görə biləsiniz. Əgər siz WordPress veb-istehsalçısı incə quraşdırmasını nəzərdən keçirmək istəyirsinizsə, onda vaxt itirmədən proqram mənbə fayllarına nəzər yetirin. WordPress-dən istifadə edərkən isə, brauzerin ünvan sətirinə diqqət edin. Xüsusilə bloqun idarə edilməsi zamanı veb proqramda olan müxtəlif PHP-fayllara müraciətləri görə biləcəksiniz. Kitabın bu fəsilində, növbəti materialların metodikasının əsasını təşkil edən məlumatlara baxılmışdır. Çox güman ki, artıq siz kiçik şəxsi PHP-proqramlar yaza bilirsiniz. Amma funksiyalara və obyektlərə həsr edilmiş növbəti fəsilə keçməzdən əvvəl aşağıdakı suallara cavab verib, mənimsədiyiniz bilikləri yoxlaya bilərsiniz.

Sual 4.1 TRUE və FALSE açar sözləri hansı verilən tipində istifadə olunur? Sual 4.2 İki ən sadə ifadə forması hansıdır? Sual 4.3 unarın, ikilinin arasında fərq və trexkomponentnımi operatorlar tərəfindən? Sual (məsələ) 4.4 nəylə operatorların şəxsi prioritetliyinin quraşdırılmasının ən yaxşı üsulunu nəticələnir (ibarətdir)? Sual (məsələ) 4.5 Operatorların ardıcıllığı anlayışı nəyi bildirir? Sual 4.6 Eynilik operatorundan (===) hansı hallarda istifadə etmək lazımdır? Sual 4.7 Şərti təlimatların üç tipini sadalayın. Sual 4.8 Hansı komanda cari təkrarlanmanı keçərək, dövrün növbəti təkrarına təkan verir? Sual 4.9 Nəyə görə for dövrü while dövrünə daha güclü dövr hesab edilir? Sual 4.10 if və while təlimatları Kimi müxtəlif məlumat tiplərindən təşkil edilmiş şərti ifadələr izah edir? Bu sualların cavabları A əlavəsinin "4-cü fəsil suallarının cavabları" bölməsində tapmaq olar.

4

PHP funksiyaları və obyektləri

İstənilən proqramlaşdırma dilinə əsas tələbləri verilənlərin saxlanılması üçün yerin mövcudluğu, proqramın və ifadələrin hesablanması, faylları idarə etmə və mətnin göstərilməsi kimi xırda predmetlərin icrası prosesinin istiqamətlənməsi vasitələrinin olmasıdır. PHP, sadaladığımız bütün kriteriyalara cavab verir.

Funksiya — konkret məsələni yerinə yetirən və əlavə olaraq hər hansı qiyməti qaytara bilən təlimatlar dəstidir. Bir neçə dəfə istifadə olunan kodun fraqmentini funksiya şəklində yerləşdirib, bu funksiyaya ad verərək kod tələb olunan yerlərdə yaratdığımız funksiyanı çağırmaq olar.

Funksiyaların fasiləsiz xətti kodu ilə müqayisədə bir çox üstünlükləri var.

- proqram mətninin hazırlanması zamanı vaxta qənaət.
- sintaktisin miqdarının və proqramlaşdırma xətalarının ixtisarı.
- proqram fayllarının yüklənməsi vaxtına qənaət.
- sürətli yerinə yetirilmə çünki, hər bir funksiya çağırılma tezliyindən asılı olmayaraq yalnız bir dəfə kompilyasiya

edilir.

- funksiyalardan istifadə etmək İmkanı necə adilərdə, belə xüsusi hadisələrdə, bir halda ki, (çünki,) onlar arqumentləri qəbul edirlər.

Obyektlər bu konsepsiyanın sonrakı inkişafıdır. ***Obyekt*** bir və ya bir neçə funksiyanı və məlumatları birləşdirir. İstifadə olunan funksiya və məlumatlar, vahid struktur yaradırlar ki, bu da ***sinif*** adlanır.

Bu fəsildə, funksiyaların təyini və çağırışı, məlumatların ötürülməsinin müxtəlif üsullarına qədər bir sıra xırda məqamlara toxunulacaq. Bu biliklərə yiyələnib, şəxsi funksiyalarınızı və obyektlərinizi (metodlarınızı) yarada və onlardan istifadə edə biləcəksiniz.

PHP funksiyaları

PHP dilinə çox zəngin imkanları olan yüzlərlə funksiya inteqrasiya edilib. Funksiyadan istifadə etmək üçün, funksiyaya ad vermək lazımdır. Məsələn, print funksiyasının necə işlədiyinə baxaq:

```
print ("print altfunksiyası");
```

PHP-də yumru mötərizələr onu bildirir ki, siz funksiyaya istinad edirsiniz. Mötərizələrin yoxluğunda hesab ediləcək ki, siz sabitə istinad edirsiniz və bu zaman da qeyri-müəyyən sabitdən istifadə haqqında xəbərdarlıq verilə bilər:

```
Notice: Use of undefined constant fname
```

Burada deyilir ki, assumed 'fname' fname-ın mətn sətiri izləyəcək, fərziyyəyə əsasən ki, siz, yəqin, koda mətn sətirini yerləşdirmək istəyirdilər. (Vəziyyət daha çox dolaşacaq (özünü itirəcək), əgər fname adında sabit əslində mövcud olacaqsa və PHP bu halda onun qiymətindən (mənasından) istifadə edəcək.)

Daha dəqiq desək, print psevdofunksiyası, adətən konstruksiya adlanır və print-dən istifadə edərkən yumru mötərizələrin qoyulması şərt deyil:

```
print "print yumru mötərizələrdən istifadəni
       tələb etmir";
```

Amma digər bir ixtiyari funksiyada hətta boş olsa belə (bu zaman, funksiyadan heç bir arqumenti ötürülmür), mötərizələrdən istifadə labüddür.

Funksiyalar sıfır daxil olmaqla istənilən miqdarda arqument qəbul edə bilər. Məsələn, aşağıda göstərilmiş phpinfo funksiyası PHP-ın cari versiyası haqqında bir çox informasiyanı özündə əks etdirir və heç bir arqument tələb etmir:

```
phpinfo();
```

phpinfo funksiyası PHP-nin cari quraşdırması haqqında informasiyanın alınması üçün olduqca faydalıdır, amma bu informasiyadan kiber-cinayətkarlar istifadə edə bilər. Buna görə də, heç vaxt şəbəkədə, iş üçün hazırlanmış kodda bu funksiyadan istifadə etməyin.

Bu funksiyanın çağırışının nəticəsi şəkildə göstərilmişdir.

Şəkil 5.1. phpinfo funksiyasının icrası nəticəsində ekranlaşdırılan informasiya

Nümunə 5.1-də bir və daha çox arqumentdən istifadə edən bir neçə inteqrasiya edilmiş funksiya göstərilmişdir.

Nümunə 5.1. Sətirlərlə iş zamanı istifadə edilən üç funksiya

```
<?php
echo strrev(" .dlrow olleH"); // Sətirinin reversivlənilməsi;
echo str_repeat("Hip ", 2);   // Sətirin təkrarı
echo strtoupper("hooray!");   // Böyük hərf registrinə çevirmə
?>
```

Bu nümunədə növbəti mətni çıxaran sətirlərin emalı üçün üç funksiya istifadə olunur:

```
Hello world.
Hip Hip
HOORAY!
```

Nəticədən göründüyü kimi, strrev funksiyası sözləri tərsinə yazır, str_repeat funksiyası Hip sətirini iki dəfə (iki arqumentinin tələbinə uyğun olaraq) təkrarlayır, strtoupper funksiyası isə "hooray!" sözünün hərflərini böyük registrə köçürür.

Funksiyanın təyini

Ümumiyyətlə funksiya təyini üçün aşağıdakı növ sintaksisdən istifadə olunur:

```
function funksiyanın_adı ([parametr [,...]])
{
  // Təlimatlar
}
```

Kvadrat mötərizələr sizi qorxutmasın, onların təyinatı barədə sonra izah edəcəyəm. Sintaksisin birinci sətirində bu göstərilmişdir:

- təyin function sözündən başlanır;
- ona, hərf və sətir xətti başlayan ad qoyulur; adın təyinində istənilən miqdarda hərf, rəqəm və ya sətir xətti nişanı qoyula bilər;
- yumru mötərizələrin mütləq mövcudluğunu nəzərə alın;
- vergüllərlə bölünmüş vacib olmayan elementə bir və ya bir neçə parametr aiddir.

Funksiyaların adlarında istifadə olunan hərflər registrə qeyri-həssasdır, buna görə də aşağıdakı bütün sətirlər eyni bir print funksiyasına istinad edəcək: `PRINT`, `Print` və `PrInT`.

Funksiya çağırışı zamanı yerinə yetirilən təlimatlar açıq fiqurlu mötərizədən etibarən başlanılır; təlimatlar birinci mötərizəni izləyən bağlı fiqurlu mötərizəylə qurtarmalıdır. Bu təlimatların tərkibində funksiyanı icrasını qurtarmağa və idarəni funksiyaya səbəb olmuş

koda qaytarmağa məcbur edən bir və ya bir neçə return təlimatı olmalıdır. Əgər return təlimatı hər hansı qiymətlə ifadə edilə bilər.

Qiymətin qaytarılması

Aşağıdakı nümunədə olan mətndəki hərf tam olaraq kiçik hərf registri ilə yazılmış formaya çevirmək lazımdır. Artıq, inteqrasiya edilmiş strtoupper PHP-funksiyası bizə tanışdır. Bizim cari funksiyamızda isə sözügedən funksiyanın əksi olan — strtolower funksiyasından istifadə ediləcək:

```
$lowered = strtolower ("HƏRF və Punktuasiya Nişanlarının siZə lazım istənilən miqdarı");

echo $lowered;
```

Bu sınağın çıxış məlumatında aşağıdakı sətir alınır:

```
hərf və punktuasiya nişanlarının sizə lazım olan istənilən miqdarı
```

Amma adlarda, soyadlarda və bu kimi digər xüsusi isimlərdə sözləri tamamilə aşağı registr hərflərindən ibarət olması düzgün deyil və biz istəyirik ki, bu tip sözlərin birinci hərfi böyük hərf registrinə uyğun yazılsın. (bu nümunədə çox da mürəkkəb olmayan, Mary-Ann və ya Jo-En-Lai kimi adlardan istifadə ediləcək.) PHP dilinin ecazkarlığı bu məsələdə də özünü biruzə verir: PHP, həmçinin ucfirst funksiyası ilə təmin olunub ki, sətirin birinci hərfini yuxarı hərf registrinə çevirir:

```
$ucfixed = ucfirst ("hərf və punktuasiya nişanlarının sizə lazım olan istənilən miqdarı");

echo $ucfixed;
```

Çıxış məlumatında aşağıdakı sətir alınır:

```
Hərf və punktuasiya nişanlarının sizə lazım olan istənilən miqdarı miqdarı
```

İndi biz bu funksiyaya proqrama daxil edə bilərik: birinci hərfi böyük hərf olan sözü almaq üçün, sətir üçün əvvəlcə `strtolower` funksiyası və sonra da ucfirst funksiyası çağırılacaq. Funksiyanın bu çağırışı üçün `strtolower` funksiyası `ucfirst` funksiya çağırışına qoyulacaq.

Bu cür hazırlanma, bizə kodun hesablanma sırasının əhəmiyyətini göstərir.

Əgər print funksiyasının aşağıdakı sadə çağırışından istifadə etsək:

```
print (5-8);
```

o halda print əvvəlcə 5-8 ifadəsini hesablayacaq və çıxışda −3 ədədi alınacaq. (Əvvəlki fəsildə göstərildiyimiz kimi, PHP nəticənin təsviri üçün nəticəni avtomatik olaraq sətirə çevirir.) Əgər ifadədə funksiya varsa, onda əvvəlcə bu funksiya hesablanır:

```
print (abs (5-8));
```

Bu qısa təlimatın icrası üçün aşağıdakı əməliyyatlar həyata keçirilir.

1. 5-8 ifadəsinin hesablanması nəticəsində −3 qiyməti alınır.
2. −3 `abs` funksiyasından (riyaziyyatda modul) istifadə edərək 3 (müsbət 3)-ə çevrilir.
3. Nəticə sətirə çevrilir və print funksiyasından istifadə edərək çevrilən sətir ekranlaşdırılır.

Belə iş prinsipinin səbəbi odur ki, PHP, ən daxilidən başlayaraq hər bir elementi hesablayır və sonda xaricdə olan əməliyyatı həyata keçirilir. Aşağıdakı funksiyaların çağırışı da analoji olaraq emal edilir:

```
ucfirst (strtolower ("HəRF və Punktuasiya Nişanlarının siZə lazım
                      istənilən miqdarı"))
```

PHP bizim koda əvvəlcə strtolower, sonra da ucfirst funksiyasını tətbiq edir və nəticədə aşağıdakı sətir:

```
Hərf və punktuasiya nişanlarının sizə lazım olan istənilən miqdarı
miqdarı
```

İndi isə biz əvvəlcə kiçik hərf, sonra isə birinci hərfi böyük registrə çevirən funksiyamızı aşağıdakı göstərilmiş üç xüsusi isimə tətbiq edəcəyik (nümunə 5.2).

Nümunə 5.2. Tam adın qaydaya salınması

```
<?php
echo fix_names ("WILLIAM", "henry", "gatES");
function fix_names ($n1, $n2, $n3)
```

```
{
    $n1 = ucfirst (strtolower ($n1));
    $n2 = ucfirst (strtolower ($n2));
    $n3 = ucfirst (strtolower ($n3));
    return $n1. "". $n2. "". $n3;
}
?>
```

İstifadəçilər adətən Caps Lock rejimini söndürməyi unudurlar. Belə olan halda isə, təsadüf böyük hərf daxil edilir. Bu nümunədəki kodunun icrası nəticəsində aşağıdakı mətn ekranlaşdırılacaq:

```
William Henry Gates
```

Massivin qayıdışı

Yuxarıda yalnız bir qiyməti qaytaran funksiyadan söhbət açılmışdır. Həmçinin, funksiyanın icra edilməsi zamanı bir neçə qiymət alma üsulları da mövcuddur. Onlardan ən təkmil üsul bu qiymətlərin massiv şəklində qaytarılmasıdır.

3-cü fəsildə göstərildiyi kimi, massiv, dəyişənlərin bir sətirə bağlanmasına oxşayır. Funksiyanın qiymətlərinin qayıdışı üçün massivdən istifadə nümunə 5.3-də göstərilmişdir.

Nümunə 5.3. Bir neçə qiymətin massiv şəklində qayıdışı

```
<?php
$names = fix_names ("WILLIAM", "henry", "gatES");
echo $names [0]. "". $names [1]. "". $names [2];
function fix_names ($n1, $n2, $n3)
{
    $n1 = ucfirst (strtolower ($n1));
    $n2 = ucfirst (strtolower ($n2));
    $n3 = ucfirst (strtolower ($n3));
    return array ($n1, $n2, $n3);
}
```

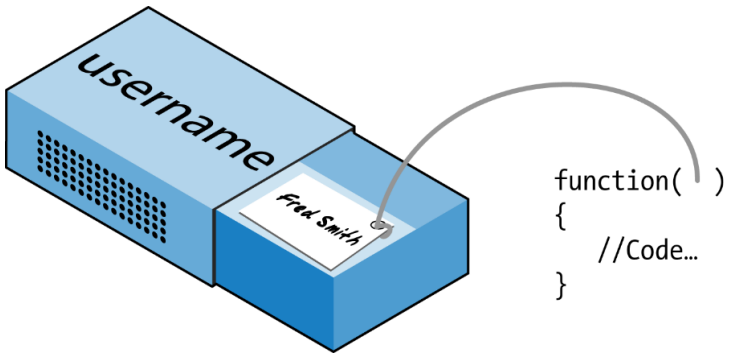
```
?>
```

Bu metodun üstünlüyü ondan ibarətdir ki, hər bir ad ayrı-ayrılıqda olur və bir sətir formasında birləşmir. Sətirdə heç bir itki olmadan istənilən istifadəçiyə onun adı və ya soyadına görə müraciət etməyə imkan verir.

İstinad üzrə ötürülmə

PHP-də dəyişənin adından əvvəl & simvolu qoyulduqda, parser başa düşür ki, dəyişənin özü deyil, istinadı ötürülür. Bu anlayış sizin üçün bir az mürəkkəb ola bilər, buna görə də 3-cü fəsildə istifadə olunan kibrit qutusu metaforasına qayıdacağıq.

Təsəvvür edin ki, qutudan kağız parçası çıxartmırsınız, amma sizdən kağızda yazılanı oxumaq, kağızdakı yazını digər bir kağıza köçürtmək, orijinal kağızı qutuya qaytarmaq, nə dəki, funksiyanın surətini çıxartmaq tələb olunmur. Siz, sadəcə sapı ilk kağız parçasına bağlayırsınız və bu sapın digər sonluğunu funksiyaya birləşdirirsiniz. İndi, funksiya müraciət olunan məlumatları tapmaq üçün, sap istiqamətində hərəkət edəcək. Beləliklə dəyişənin bütün xərcləri yalnız nəzərdə tutulmuş surətə xaric edilir (istisna edilir) ki, funksiyada (rolda) onun qiymətindən (mənasından) istifadə etmək mümkün olsun. Bundan başqa, indi funksiya dəyişənin qiymətini dəyişdirə bilər. Deməli, nümunə 5.3 belə köçürtmək olar: bütün parametrlərə istinadlar vermək lazımdır ki, bundan sonra funksiya dəyişiklikləri (nümunə 5.4) birbaşa daxil edə bilsin.



Şəkil. 5.2. İstinadın Dəyişənin qiymətinə bağlanmış sap şəklində təsəvvürü

Nümunə 5.4. Funksiyadan qiymətlərin istinad üzrə qayıdışı

```
<?php
  $a1 = "WILLIAM";
  $a2 = "henry";
  $a3 = "gatES";
  echo $a1. "". $a2. "". $a3. "<br>";
  fix_names ($a1, $a2, $a3);
  echo $a1. "". $a2. "". $a3;
  function fix_names (&$n1, &$n2, &$n3)
  {
    $n1 = ucfirst (strtolower ($n1));
    $n2 = ucfirst (strtolower ($n2));
    $n3 = ucfirst (strtolower ($n3));
  }
?>
```

Sətirlərin ötürülməsi zamanı funksiyalar əvvəlcə bilavasitə sətirləri qiymətlər şəklinlərə dəyişənlərə mənimsədir və ekrana çıxarılır. Sonra, əvvəlki qaydada, funksiya çağırılır, amma indi hər parametrin adından əvvəl & simvolu qoyulur ki, funksiya yalnız dəyişənlərin qiymətlərinə istinad etsin.

İndi $n1, $n2 və $n3 dəyişənlərini $a1, $a2 və $a3 dəyişənlərinin qiymətlərinə aparan "saplar" bağlanmışdır. Başqa sözlə, qiymətlərin bir qrupu mövcuddur ki, amma onlara dəyişənlər və icazə verilən adlar kimi iki formada müraciət etmək olur.

Buna görə də fix_names funksiyaları yalnız $a1, $a2 və $a3 dəyişənlərinin qiymətlərini yeniləmək üçün yeni qiymətləri $n1, $n2 və $n3 dəyişənlərinə mənimsədəcək. Bu kodun icrası nəticəsində aşağıdakı sətirlər alınacaq:

```
WILLIAM henry gatES
William Henry Gates
```

Gördüyünüz kimi, hər iki echo təlimatında yalnız $a1, $a2 və $a3 dəyişənlərinin qiymətlərindən istifadə olunur. İstinad üzrə

qiymətlərin ötürülməsi zamanı ehtiyatlı olmaq lazımdır. Əgər ilkin mənanı saxlamaq lazımdırsa, dəyişənlərinizin surətini çıxarın, sonra da bu surətlərin qiymətlərini istinad üzrə verin.

Qlobal dəyişənlərin qayıdışı

Onun düz qlobal funksiyanın bədənindən elan edib həmçinin funksiyanı onun hüdudları xaricində dəyişənə, yaradılmışa girişi vermək Olar.

Qlobal dəyişənlərin qayıdışı onun düz qlobal funksiyanın bədənindən elan edib həmçinin funksiyanı onun hüdudları xaricində dəyişənə, yaradılmışa girişi vermək Olar.

Onun düz qlobal funksiyanın bədənindən elan edib həmçinin funksiyanı onun hüdudları xaricində dəyişənə, yaradılmışa girişi vermək olar. Bu tip dəyişənlərin adı global açar sözü ilə müşayiət olunmalıdır. Bu tip dəyişənlərə kodunuzun istənilən hissəsindən giriş almaq mümkün olacaq (nümunə 5.5).

Nümunə 5.5. Qlobal dəyişənlərdə qiymətlərin qayıdışı

```
<?php
  $a1 = "WILLIAM";
  $a2 = "henry";
  $a3 = "gatES";
  echo $a1. "". $a2. "". $a3. "<br>";
  fix_names(); echo $a1. "". $a2. "". $a3;
  function fix_names()
  {
    global $a1;
    $a1 = ucfirst (strtolower ($a1));
    global $a2;
    $a2 = ucfirst (strtolower ($a2));
    global $a3;
    $a3 = ucfirst (strtolower ($a3));
  }
?>
```

İndi, funksiyaya parametrlər vermək məcburi deyil, çünki artıq parametrlərə ehtiyac yoxdur. Bu tip dəyişənlərin elanın sonra, bu dəyişənlər funksiyalar daxil olmaqla proqram kodunun bütün hissələrində qlobal və əlçatan olur.

Saxlanma üçün kimi dəyişənlərin ən böyük lokal görünüşünə olar massivləri qaytarmaq və ya istinad (sürgün) üzrə (görə) dəyişənlərdən, verilmişlərdən (çatdırılmışlardan) istifadə etmək lazımdır. Əks təqdirdə, funksiyalar bəzi üstünlüklərini itməyə başlayacaqlar.

Dəyişənlərin görünmə sahəsi haqqında

Bu barədə artıq 3-cü fəsildə bəhs edilmişdir. Ona görə qısa xatırlatma edəcəyəm.

- Lokal dəyişənlər kodun müəyyən edildiyi hissəsində əlçatandır. Əgər lokal dəyişən funksiya xaricində olarsa, dəyişənlərə giriş funksiyalar, siniflər və s xaric bütün koddan mümkün olacaq Əgər dəyişən funksiyanın daxilində müəyyən edilmişsə, deməli, bu dəyişənə yalnız funksiya kodu daxilində müraciət edilə bilər və bu dəyişənin qiyməti funksiyadan çıxış anında itir.
- Qlobal dəyişənlər kodunuzun istənilən hissəsində əlçatandır.
- Statik dəyişənlər yalnız elan edildiyi funksiyanın daxilində əlçatandır, amma bu halda bu tip dəyişənlər çoxqat funksiya çağırışları prosesində öz qiymətini saxlayır.

Faylların əlavə edilməsi və sorğusu

PHP proqramlaşdırma bacarıqlarına yiyələnərkən, ehtimal ki, funksiyalardan ibarət olan kitabxana yaradacaqsınız. Bundan başqa, yəqin ki, digər proqramçılar tərəfindən yaradılmış kitabxanalardan da istifadə etməyə başlayacaqsınız.

Bu funksiyaları kopyalamaq və onları birbaşa olaraq öz kodunuza daxil etmək məntiqsizlikdir. Bu funksiyaları ayrı-ayrı fayllarda saxlamaq və komandalardan istifadə edərək, bu fayllara istinad etmək olar. Bunun üçün iki — `include` (daxil etmək) və `require` (tələb etmək) komandaları mövcuddur.

include təlimatından istifadə

include təlimatı konkret olaraq, faylın çıxardılması və bu faylın bütün tərkibinin yükləməsinə şərait yaradır. Bu əməliyyat cari faylın daxilinə istinad edilən faylın tərkibinin əlavə edilməsi əməliyyatına ekvivalentdir. Nümunə 5.6-də library.php adlı faylın necə daxil edilməsi göstərilir.

Nümunə 5.6. PHP faylın cəlb edilməsi

```
<?php
  include "library.php";
  //Burada sizin kodunuz yerləşir
?>
```

include_once təlimatından istifadə

`include_once` təlimatı hətta tələb edilən fayl artıq yerləşdirilmişsə belə, faylı təkrarən daxil edir. Fərz edək ki, *`library.php`* kitabxanasında bir çox faydalı funksiyalar mövcuddur. Siz bu kitabxananı və bu kitabxananı (*library.php*) ehtiva edən digər bir kitabxananı include təlimatının köməyilə öz faylınıza daxil edirsiniz. Bu qoyuluşa görə siz library.php-ni iki dəfə daxil edirsiniz və bu yolverilməz haldır. Nəticədə xəta barədə mesajlar meydana çıxacaq, çünki eyni sabiti və ya funksiyanı bir neçə dəfə elan etmək olmaz. Buna görə də include təlimatının yerinə `include_once` təlimatından istifadə etmək lazımdır (nümunə 5.7).

Nümunə 5.7. PHP faylının bir neçə dəfə qoşulması

```
<?php
  include_once "library.php";
  //Burada sizin kodunuz yerləşir
?>
```

Əgər artıq yerinə yetirilmiş oxşar kitabxana ehtiva edən include və ya `include_once` təlimatı varsa, bu təlimatlara əhəmiyyət verilməyəcək. Faylın artıq qoşulmuş olduğunu müəyyən etmək üçün, include təlimatında göstərilmiş fayla aparan mütləq yolu, yolda tapılmış bütün açılmış nisbi yollar və fayllar ilə müqayisə etmək lazımdır.

Ümumiyyətlə, hər bir halda yaxşı olar ki, include_once təlimatından istifadə edəsiniz. Bu halda, include təlimatında yaranan xətalardan sığortalanacaqsınız.

require və require_once təlimatlarından istifadə

`include` və `include_once` təlimatlarından istifadə zamanı yaranan potensial problem ondan ibarətdir ki, PHP lazımlı fayla yalnız bircə dəfə sorğu göndərəcək. Hətta göstərilən fayl tapılmadıqda belə proqramın icrası davam edəcək.

Yerləşdirilən fayl prinsipial əhəmiyyətə malikdirsə, faylı tələb etmək, yəni require təlimatını tətbiq etmək lazımdır. Yuxarı qeyd edilən include_once təlimatının üstünlükləri analoji olaraq require_once təlimatına da aiddir. Ona görə də məsləhət görürəm ki, faylın tələb edilməsinə ehtiyac yarandıqda, require_once təlimatından istifadə edin (nümunə 5.8).

Nümunə 5.8. PHP faylının bir neçə dəfə tələb edilməsi

```
<?php
require_once "library.php";
//Burada sizin kodunuz yerləşir
?>
```

PHP versiyalarının uyğunluğu

PHP daim təkmilləşməşir və bu təkmilləşdirmələr versiyada şəklində təqdim edilir. Əgər konkret olaraq hər hansı funksiyanın kodunuzda mümkünlüyünü yoxlamaq lazımdırsa, function_exists funksiyasından istifadə etmək olar. Bu funksiya sorğu edilən

funksiyanı funksiyanın qabaqcadan müəyyən edilmiş və yaradılmış funksiyalar siyahısında yoxlayır.

Nümunə 5.9-da PHP 5 versiyasında mövcud olan array_combine funksiyasının mümkünlüyü yoxlanılır.

Nümunə 5.9. Funksiyanın mövcudluğunun yoxlanılması

```
<?php
  if (function_exists (" array_combine"))
  {
    echo "Funksiya mövcuddur";
  }
  else
  {
    echo "Funksiya mövcud deyil. Funksiyanın müstəqil yaradılması
          istəyə bağlıdır";
  }
?>
```

Bu cür koddan istifadə edərək PHP-nin yeni versiyalarında olan istənilən funksional imkanlarından istifadə etmək olar. Əgər istədiyiniz funksiya istifadə etdiyiniz PHP versiyasında mövcud deyilsə, siz mümkün funksiyalardan istifadə edib bu funksiyanın modelini hazırlamaya bilərsiniz. Sizin yazdığınız funksiyalar daha yavaş işləsə də, ən azı kodunuz daha geniş mobilliyə malik olacaq.

İstifadə etdiyiniz PHP versiyasını müəyyən etmək üçün `phpversion` funksiyasından istifadə etmək olar. Versiyadan asılı olaraq aşağıdakı növdə nəticə alınacaq:

```
5.5.11
```

PHP obyektləri

Funksiyaların tətbiqi ilə praktik olaraq, hesablama texnikasının inkişafı (ən elementar təlimatlar sayılan GOTO və ya GOSUB o zamanlar, əlçatan ən yaxşı proqram naviqasiyası vasitələri idi), proqramlaşdırmanın effektivliyini əhəmiyyətli dərəcədə artırdı. *Obyekt yönümlü proqramlaşdırmanın* (***OYP***) inkişafı ilə funksiyalar, tamamilə yeni səviyyəyə qalxdı.

Kodun təkrarən istifadə edilən fraqmentlərini funksiya şəklində təyin etmək bacarığınızı biraz da təkmilləşdirib, funksiyaları və məlumatları bir-birilə əlaqələndirməyi öyrədəcəyik. Belə əməliyyat *obyekt* adlanır.

Bir çox müxtəlif hissədən ibarət olan sosial şəbəkə saytlarına nəzər yetirək. Belə hissələrdən biri, istifadəçi tabeliyində olan bütün funksiyaları idarə etməsidir: bu kod nəticəsində istifadəçi qeydiyyatdan keçərək öz məlumatlarını daxil edə, artıq qeydiyyatdan keçmiş istifadəçilər isə öz şəxsi məlumatlarını dəyişdirə bilər. Bütün bu əməliyyatları idarə etmək üçün standart PHP-də bir sıra funksiyalar yaratmaq və bütün istifadəçilər üzrə məlumatların yoxlanması və mübadiləsi üçün MySQL verilənlər bazasına bir neçə sorğu göndərmək lazımdır.

İndi təsəvvür edin ki, bütün bunlar obyekt şəklində olsa, əməliyyatlar nə qədər sadə olacaq. Bunun üçün, bütün kodu istifadəçilərə xidmət üçün və sinifin daxildə olanan məlumatlarla, dəyişənlərlə işləmək üçün nəzərdə tutulmuş User adlı sinif yaratmaq olar. Bundan sonra istifadəçi məlumatlarını idarə etmək istədikdə, sadəcə olaraq `User` sinifinin yeni obyekti yaradılacaq.

Bu yeni obyektə cari istifadəçi kimi baxmaq mümkün olacaq. Məsələn, obyektə ad, parol və elektron poçt ünvanını vermək, obyektdən bu adda istifadəçinin olduğunu soruşmaq və əgər həqiqətən də bu adda istifadəçi yoxdursa, bu atributlar vasitəsilə yeni istifadəçinin qeydiyyatı aparılacaq. Hətta ani mesajların mümkünlüyü təmin edən obyekt və ya iki istifadəçi arasında dostluq münasibətlərini qurmağa icazə verən obyekt yaratmaq olar.

Terminologiya

Obyektlərdən istifadə edən proqramın yaradılması Belə Məlumatlar, kod yığımı sinif adlandırılır. Bu sinifə əsaslanan hər yeni obyekt bu sinifin nüsxəsi adlanır (və ya istifadə hadisəsi).

Obyektlə bağlı məlumatlar, xüsusiyyət və bu xüsiyyətlərdən istifadə edən funksiyalar isə metodlar adlanır. Sinifin təyini zamanı onun xüsusiyyətlərinin və metodlarının adları koda əlavə edilir. də obyektin metaforası musiqi avtomatı şəklində göstərilmişdir. Karuseldəki kompakt-disklərə xüsusiyyət kimi, disklərin işə

salınmasına səbeb olan ön paneldəki düyməyə metod kimi baxmaq olar. Həmçinin düymələrin (metod, obyektin aktivləşməsi üçün istifadə edilən) basılması və kompakt-disklərin (musiqinin çıxardılması üçün istifadə edilən metod və ya xüsusiyyətlər, kompakt-disklər rolunda) oxunması qurğusu üçün çat var.

Obyektlərin inkapsulativ (paketləşdirilmiş) yaradılması məqsədəuyğundur. Belə olduqda, xarici kodun obyektin verilənlərinə girişi qadağan edilir. Obyektlə verilən metodlar obyektin interfeysini təşkil edir.

Belə yanaşma düzəlişləri sadələşdirir: qüsurlu kodu yalnız sinifin daxilində düzəltmək lazım olacaq. Bundan başqa, eyni inkapsulyasiyadan və interfeys dəstəyindən istifadə etmək nəticəsində proqramın yenilənməsi zamanı yeni siniflər hazırlamaq, köhnə siniflərin tamamilə qaydaya salınması və sonra müvafiq siniflər ilə əvəz edilməsi mümkün olacaq. Əgər onlar yeni sinifin iş prosesində xəta yaranarsa, sinifə düzəlişlər edənə qədər sabit köhnə versiyanı bərpa edə bilərsiniz.

Sinif yaradılan kimi, aydın olur ki, sinifi oxşayar, amma yenə də fərqlənən sinifə ehtiyac var. Varislikdən istifadə, hər bir yeni sinifi daha sürətli və daha sadə müəyyən edəcək. Bu halda sizin yeni sinifiniz varisinə məxsus bütün xüsusiyyətləri saxlayacaq. İndi, ilk sinif supersinif, və yeni sinifiniz isə yarımsinif (və ya törəmə sinif) adlandırılacaq.

Şəkil. 5.3. Avtonom obyektin musiqi avtomatı ilə uyğunluğu

Məsələn musiqi avtomatına qayıdaq. Əgər siz musiqi ilə yanaşı videoların da icrasını təmin edən yeni musiqi avtomatını ixtira edirsinizsə onda bu musiqi avtomatına ilk musiqi avtomatının bütün xüsusiyyətlərini və metodlarını saxlamaqla, sadəcə bir neçə yeni xüsusiyyət (videokliplər) və yeni metod (videopleerov) əlavə edə bilərsiniz.

Bu sistemin əhəmiyyətli üstünlüyü ondan ibarətdir ki, əgər siz işin sürətini artırırsınızsa və ya supersinifin işinin başqa aspektlərini yaxşılaşdırırsınızsa, bu supersinifin yarımsinifləri ən yeni təkmilləşdirmələrə malik olur.

Sinifin elanı

obyektin istifadə etmək fürsətini almazdan əvvəl, class açar sözünün köməyi ilə sinifi müəyyən etmək lazımdır. Sinif özündə ad (hərflər registrə həssasdır), xüsusiyyətləri və metodlar ehtiva edir. Nümunə 5.10-da $name və $password xüsusiyyətinə malik `User` sinifinin təyin edilməsi göstərilir (hansı ki, public açar sözüylə göstərilmişdir — bu barədə daha ətraflı məlumat almaq üçün cari bölmənin "PHP 5-də xüsusiyyətlərin və metodların görünmə sahəsi" yarımfəsilinə baxmaq lazımdır). Həmçinin bu zaman sinifin yeni nüsxəsi ($object adında) yaradılır.

Nümunə 5.10. Sinifin elanı və obyektin yoxlanılması

```
<?php
  $object = new User;
  print_r ($object);
  class User
  {
    public $name, $password;
    function save_user()
    {
      echo "Burada istifadəçinin məlumatlarını ehtiva edən kod yerləşir";
    }
  }
?>
```

Burada həmçinin birbaşa olaraq, print_r adlı qiymətsiz funksiya cəlb edilmişdir. O PHP-dan götürmə üçün rahatda dəyişən haqqında informasiyanı insan tərəfindən formaya əks etdirməyi tələb edir, nə haqqında onun adında (bildirən readable — "oxunan") _r elementini danışır (deyir). bu funksiya $object-in yeni obyekti üçün aşağıdakı informasiyanı çıxarır:

```
User Object
(
  [name] =>
  [password] =>
)
```

Amma brauzer bütün boş sahələri sıxır, buna görə də çıxarılan informasiya bir az mürəkkəb oxunur:

```
User Object ([name] => [password] =>)
```

Hər halda çıxarılmış informasiya onu göstəririr ki, $object istifadəçi tərəfindən müəyyən edilmiş obyektdir və name, password xüsusiyyətlərini özündə ehtiva edir.

Müəyyən sinif obyektinin yaradılması

Obyektin yaradılması üçün `new` açar sözündən istifadə olunur: `obyekt = new Sinif`. Obyekt yaradılmasının iki üsulu mövcuddur:

```
$object = new User;
$temp = new User ('name', 'password');
```

Birinci sətirdə biz sadəcə obyekti User sinifinə təyin edirik. Amma ikinci sətirdə parametrləri də veririk.

Sinif arqumentləri tələb edə və ya qadağan edə bilər; həmçinin də arqumentlərə icazə verə, onları tələb etməyə bilər.

Obyektlərə giriş

Nümunə 5.10 bir qədər də sətirlər əlavə edək və nəticəni yoxlayaq. Əvvəlki kod nümunədə 5.11-də, obyektin xüsusiyyətlərinin quraşdırılması və metodun çağırışı hesabına genişlənir.

Nümunə 5.11. Obyektin yaradılması və yaradılan obyektlə qarşılıqlı təsir

```
<?php
  $object = new User;
```

```
    print_r ($object);
    echo "<br>";
    $object->name = "Joe";
    $object->password = "mypass";
    print_r ($object);
    echo "<br>";
    $object->save_user();
    class User
    {
      public $name, $password;
      function save_user()
      {
       echo "Burada istifadəçinin məlumatlarını ehtiva edən kod yerləşir";
      }
    }
  ?>
```

Nümunədən göründüyümü kimi, obyektin xüsusiyyətinə giriş üçün aşağıdakı sintaksisdən istifadə olunur: *$obyekt->xüsusiyyət*. Oxşar üsulla metodu da çağırmar olar: *$obyekt>metod().*

Qeyd etmək lazımdır ki, xüsusiyyət və metodların adlarından əvvəl dollar simvolları ($) qoyulmur. Əgər adların əvvəli $ simvolu qoyularsa, onda kod işləməyəcək, çünki, bu zaman dəyişəndə saxlanılan qiymətə müraciət etməyə cəhd edəiləcək. Məsələn, `$object->$property` ifadəsi `$property` adlı dəyişənə mənimsənmiş qiyməti (tutaq ki, bu qiyməti brown sətiridir) tapmağa çalışacaq, sonra da `$object->brown` xüsusiyyətinə müraciət edəcək. Əgər `$property` dəyişəni müəyyən edilməmişsə, onda `$object->NULL` xüsusiyyətinə müraciət etmək cəhdi göstərəcək və bu zaman da xəta baş verəcək.

Əgər brauzerdə olan mənbə koduna baxsaq, onda nümunə 5.11 mənbə kodu aşağıdakı şəkildə olacaq:

```
User Object
(
 [name] =>
```

```
 [password] =>
)
User Object
(
 [name] => Joe
 [password] => mypass
)
Burada istifadəçinin məlumatlarını ehtiva edən kod yerləşir
```

Burada da həmçinin print_r funksiyası istifadə olunur, hansı ki, mənimsəmədən əvvəl və sonra $object dəyişəninin tərkibini qiymətin (mənanın

Burada da həmçinin mənimsətmədən əvvəl və sonra $object dəyişəninin tərkibini qiymətin xüsusiyyəti şəklində verən print_r funksiyasında istifadə olunur. Gələcəkdə mən daha print_r təlimatından istifadə etməyəcəyəm və əgər bu kitabın materialı sizin hazırlanma serverinizə uyğundursa, siz koda tam təqdimata malik bir neçə belə təlimat yerləşdirə biləcəksiniz.

Həmçinin qeyd etmək olar ki, `save_user` metodunun çağırışı sayəsində bu metodun kodu yerinə yetirilmişdi, hansı ki, sətiri çıxardı, onu xatırladanı ki, hansısa kodu yaratmaq lazımdır.

Funksiyaları və sinifləri istifadə olunan təlimatdan əvvəl və ya sonra kodunuzun istənilən yerində yerləşdirə bilərsiniz. Amma "qəbul olunmuş proqramlaşdırma qaydası"na uyğun olaraq funksiyaları və sinifləri kodun sonuna yerləşdirməyinizi məsləhət görürəm.

Obyektlərin klonlaşdırılması

Əgər obyekt artıq yaradılmışsa, onda parametr kimi o istinad üzrə ötürülür. Əgər kibrit qutusunun metaforasından istifadə etsək, onda bu görünür bağlanmaya birdən (dərhal) obyektə bir neçə (bir qədər) sap, qutuda saxlanılana, nə bağlanmış saplardan hər hansıya görə izləyərək (gedərək) ona giriş əldə etməyə icazə verir. Başqa sözlə, obyektlərin mənimsənməsi onların tam kopyalamasına səbəb olmur.

Bunun necə işlədiyi, nümunə 5.12-də göstərilmişdir. Burada User-in heç bir metoda malik olmayan və yalnız bir xüsusiyyət, name xüsusiyyətini ehtiva edən çox sadə istifadəçi sinifi təyin edilir.

Nümunə 5.12. Obyektin kopyalanması

```
<?php
  $object1 = new User();
  $object1->name = "Alice";
  $object2 = $object1;
  $object2->name = "Amy";
  echo "object1 name = ". $object1->name. "<br>";
  echo "object2 name = ". $object2->name;
  class User
  {
    public $name;
  }
?>
```

Biz $object1 obyektini yaratdıq və name xüsusiyyətinə Alice qiymətini mənimsətdik. Sonra `$object2` yaradıldı və ona `$object1`-in qiymətini mənimsədildi. Bundan sonra, Amy qiyməti bilavasitə `$object2` obyektinin name xüsusiyyətinə mənimsədildi. Amma bu kod aşağıdakı informasiyanı ekranlaşdıracaq:

```
object1 name = Amy
object2 name = Amy
```

Niyə belə oldu? Çünki, `$object1` və `$object2` obyekti bir-birinə istinad edir, buna görə də `$object2`-ə aid olan name xüsusiyyətinin dəyişikliyi, $object1-ə aid olan müvafiq xüsusiyyətə də təsir edir. Belə qarışıqlıq yaranmasın deyənə clone təlimatından istifadə etmək lazımdır. Bu təlimatın icrası zamanı sinifin yeni nüsxəsi yaradılır və yeni nüsxəyə ilk sinifin xüsusiyyətlərinin qiymətləri köçürülür. Bu təlimatın tətbiqi nümunə 5.13-də göstərilmişdir.

Nümunə 5.13. Obyektin klonlaşdırılması

```
<?php
  $object1 = new User();
  $object1->name = "Alice";
```

```
$object2 = clone $object1;
$object2->name = "Amy";
echo "object1 name = ".
$object1->name. "<br>";
echo "object2 name = ". $object2->name;
class User
{
  public $name;
}
?>
```

Buda belə. Bu kod, bizə lazım olan nəticəni verir:

```
object1 name = Alice
object2 name = Amy
```

Konstruktorlar

Vaxtı konstruktorlar səbəb olunan (çağırılan) sinifə arqumentlərin siyahısını vermək (çatdırmaq) olar. Onlar sinifin daxilində xüsusi metoda ötürülürlər ki, bu metod konstruktor adlanır və müxtəlif xüsusiyyətlərin inisializasiyasıyla məşğul olur.

Nümunə 5.14. Metod-konstruktorun yaradılması

```
<?php
  class User
  {
  function User ($param1, $param2)
  {
    // Burada konstruktor təlimatları yerləşir
    public $username = "Guest";
  }
  }
?>
```

Nümunə 5.15-də göstərilmişdir ki, PHP 5 konstruktor adına mənimsətmə zamanı daha məntiqli yanaşma təqdim edir. Bu zaman funksiya adına __construct frazası mənimsədilir.

Nümunə 5.15. PHP 5-də metod-konstruktorun yaradılması

```
<?php
```

```
  class User
  {
  function __construct ($param1, $param2)
    {
      // Burada konstruktor təlimatları yerləşir
      public $username = "Guest";
    }
  }
? >
```

PHP 5-də destruktorlar

PHP 5-də metod-destruktorların yaradılması imkanı daha bir yenilik sayılır. Bu imkan, kod sonuncu dəfə obyektə istinad edərkən və ya ssenari sona yaxınlaşan zaman işə yarıyır. Nümunə 5.16-da, metod-destruktorun yaradılması göstərilmişdir.

Nümunə 5.16. PHP 5-də metod-destruktorun yaradılması

```
<?php
  class User
  {
   function __destruct()
   {
   // Burada destruktor kodu yerləşir
   }
  }
?>
```

Metodların yazılışı

Gördüyünüz kimi, metodun elanı, funksiyanın elanına bənzəyir, amma bəzi fərqlər mövcuddur. Məsələn, cüt sətir xətti (__) ilə başlanan metodların adları ehtiyata saxlanılan sözdür və siz bu adda heç nə yarada bilməzsiniz.

Bundan başqa, cari obyektin xüsusiyyətlərinə giriş üçün istifadə oluna bilən xüsusi `$this` dəyişəni mövcuddur. Bunun iş prinsipini anlamaq üçün nümunə 5.17-dəki koda baxın. Kodda User sinifinin

təyinini ehtiva edən və get_password adlanan daha bir metod mövcuddur.

Nümunə 5.17. $this dəyişəni metodundan istifadə

```
<?php
  class User
  {
    public $name, $password;
    function get_password()
    {
    return $this->password;
    }
  }
?>
```

parolun alınması metodu — `get_password` — cari obyektə giriş üçün $this dəyişəni tətbiq edilir və sonra da bu obyektə aid olan password xassəsinin qiyməti qaytarılır. Nəzərə alın ki, - > operatorundan istifadə zamanı $password xüsusiyyətinin adındakı ilk simvol olan $ simvolu ixtisara düşür. Əgər bu xüsusiyyət, xüsusilə bu xüsusiyyətin ilk tətbiqi zamanı əvvəlki qaydasında daxil edilsə, olduqca tipik səhv buraxılacaq.

Nümunə 5.17-də müəyyən edilmiş sinifi aşağıdakı qaydada istifadə etmək lazımdır:

```
$object = new User;
$object->password = "secret";
echo $object->get_password();
```

Bu kod secret parolunu çıxarır.

PHP 5-də statik metodlar

PHP 5-də işləmək Metodları, həmçinin statik metod kimi də müəyyən etmək olar. Bu, metodun obyektdə deyil, sinifdə çağrılması imkanını bildirir. Statik metod, obyektin heç bir giriş xüsusiyyətinə malik deyil. Belə metodun yaradılması və yaradılmış metoda giriş nümunə 5.18-də göstərilmişdir.

Nümunə 5.18. Statik metodun yaradılması və ona giriş

```
<?php
  User::pwd_string();
  class User
  {
   static function pwd_string()
   {
    echo "Zəhmət olmasa, öz parolunuzu daxil edin";
   }
  }
?>
```

Nəzərə alın ki, statik metodla yanaşı özü sinifi çağırılır və necə bu halda operatorun yerinə - > ikiqat iki nöqtənin operatoru istifadə olunur (::), həmçinin görünmə sahəsinin icazəsinin operatoru kimi məlum (məşhur). Statik funksiyalar sinifin konkret nüsxələrinə deyil, birbaşa sinifə aid olan əməliyyatların edilməsi üçün faydalıdır. Statik metoddan istifadə nümunə 5.21-də sonra göstərilmişdir.

`$this>property` *ifadəsinin köməyilə cari obyektin xüsusiyyətinə giriş əldə etmək və ya statik funksiyanın daxilində obyektin başqa xüsusiyyətlərinə giriş əldə etmək cəhdinin xətası barədə məlumat veriləcək.*

Siniflərin daxilində xüsusiyyətlərin açıq elanına ehtiyac yoxdur, çünki, onların ilk istifadəsi naməlum obrazla müəyyən edilə bilər. Bu xüsusiyyət illüstrasiyası, nümunə 5.19-da göstərilmiş User sinifi nə xüsusiyyətə nə də metod malik deyil, amma sinifin kodunun təyinində qeyri-qanuni heç nə yoxdur.

Nümunə 5.19. Xüsusiyyətin naməlum elanı

```
<?php
  $object1 = new User();
  $object1->name = "Alice";
  echo $object1->name;
  class User {}
```

```
?>
```

Bu kod tamamilə düzgün və problemsiz Alice sətirini ekranlaşdıracaq, çünki, PHP sizin üçün `$object1->name` dəyişənini naməlum obrazla elan edəcək. Amma belə proqramlaşdırma stili tapılması çətin olan səhvlərə gətirib çıxara bilər. Səhvin tapılma çətinliyinin səbəbi odur ki, name xüsusiyyəti sinifin xaricində elan edilmişdir.

Özünüzə əlavə problem yaratmamaq üçün məsləhət görürəm ki, istənilən halda sinifin daxilində olan xüsusiyyətləri aşkar formada elan edin. İnanın ki, bu halda belə problemlərlə qarşılaşmayacaqsınız.

Bundan başqa, xüsusiyyət sinifin daxilində elan edildikdə, ona susmaya görə qiymət mənimsətmək olar. Tərəfinizdən istifadə edilən qiymət sabit olmalıdır və funksiya çağırışı və ya ifadənin hesablanması nəticəsinə səbəb olmamalıdır. Bir neçə mümkün və yolverilməz mənimsətmə formaları nümunə 5.20-də göstərilmişdir.

Nümunə 5.20. Xüsusiyyətlərin mümkün və yolverilməz elanı

```
<?php
  class Test
  {
    public $name = "Paul Smith"; // Mümkün
    public $age = 42;            // Mümkün
    public $time = time();       // Yolverilməz –xətalı çağırış
    public $score = $level * 2;  // Yolverilməz – xətalı ifadə
  }
?>
```

Funksiyaların təyininin daxilində yaradılan qlobal sabitlər, analoji olaraq, siniflərin daxilində də müəyyən edilir. Sabitlər ümumi fonda seçilməlidir və adətən onların adlarında böyük hərf registrindən istifadə olunur (nümunə 5.21).

Nümunə 5.21. Sinifin daxilində sabitlərin təyini

```
<?php
  Translate::lookup();
  class Translate
  {
    const ENGLISH = 0;
```

```
        const SPANISH = 1;
        const FRENCH = 2;
        const GERMAN = 3;
        //...
        static function lookup()
        {
          echo self::SPANISH;
        }
      }
    ?>
```

sabitlərə self açar sözünün və cüt iki nöqtə operatorunun köməyi ilə birbaşa olaraq müraciət etmək olar. Nəzərə alın ki, kodun birinci sətirində istifadə edilən cüt iki nöqtə operatoru, sinifin ilkin nüsxəsi olmadan birbaşa olaraq çağırışını həyata keçirir. Gözlənildiyi kimi, kodun icrası nəticəsində göstərilən qiymət 1-ə bərabər olacaq. Yadda saxlayın ki, sabit, təyinindən sonra dəyişdirilə bilməz.

PHP 5-də xüsusiyyətlərin və metodların görünmə sahəsi

PHP 5-də xüsusiyyətlərin və metodların görünmə sahəsinin idarə edilməsi üçün üç açar sözü təklif edilir:

`public` (açıq).

Bu görünmə sahəsindəki xüsusiyyətlər public açar sözü ilə yaradılır. Həmçinin dəyişənin elanı zamanı və ya gizli dəyişənin ilk istifadəsi zamanı susmaya görə təyin edilir. var və public açar sözləri bir-birilə qarşılıqlı əvəz oluna bilər. Hərçənd ki, hal-hazırda var-dan istifadə o qədər də dəstəklənmir və bu açar söz PHP-nin əvvəlki versiyaları ilə uyğunluğun qorunması üçün saxlanılmışdır. Susmaya görə metodlar açıq hesab edilir;

`protected` (qorunmuş).

Bu görünmə sahəsindəki xüsusiyyətlərə və metodlara yalnız obyektlərə aid olan sinif metodları və ixtiyari yarımsiniflərin eyni metodları vasitəsilə istinad etmək mümkündür;

`private` (qapalı).

Bu görünmə sahəsində olan sinifin nümayəndələrinə, yalnız bu sinif daxilində olan metodlar vasitəsilə müraciət etmək olar.

Hansı görünmə sahəsini tətbiq etmək qərarı aşağıdakı vəziyyətlər əsasında qəbul edilir:

- sinifin nümayəndəsinə xarici koddan girişə və siniflərin genişləndirilmiş varisliyinə ehtiyac olacaqsa açıq (public) görünmə sahəsi tətbiq etmək lazımdır;
- sinifin nümayəndəsinə xarici koddan girişə qadağan etmə, lakin siniflərin genişləndirilmiş varisliyi icazə vermə qorunmuş (protected) görünmə sahəsi istifadə etmək lazımdır;
- sinifin nümayəndəsinə heç bir xarici koddan giriş və siniflərin genişləndirilmiş varisliyi olmayan şəraitdə qapalı (private) görünmə sahəsi tətbiq etmək lazımdır.

Bu açar sözlərin tətbiqi nümunə 5.22-də göstərilmişdir.

Nümunə 5.22. Xüsusiyyətin və metodun görünmə sahəsinin dəyişikliyi

```
<?php
  class Example
  {
  var $name = "Michael"; //Tövsiyə edilməyən forma
  public $age = 23; // Açıq xüsusiyyət
  protected $usercount; // Qorunmuş xüsusiyyət
  private function admin() //Qapalı metod
  {
   // Bura adminstrasiyaya uyğun qapalı kod yerləşir
  }
  }
?>
```

Statik xüsusiyyətlər və metodlar

Əksər məlumatlar və metodlar sinifin nüsxələrində tətbiq edilir. Məsələn, User sinifində istifadəçinin konkret parolunu təyin etmək və ya artıq üzv olmuş istifadəçinin qeydiyyatının yoxlanılması lazımdır. Bu faktlar və əməliyyatlar hər bir konkret istifadəçiyə

xüsusi münasibət tələb edir və buna görə də xüsusiyyətin və metodların nüsxəsi üçün spesifiklər tətbiq edilir.

Amma vaxtaşırı aid olan bütün məlumatlara ehtiyac yarandıqda, bütün sinifin xidmət göstərməsi lazımdır. Məsələn, saytda qeydiyyatdan keçmiş istifadəçinin nə qədər olduğunu öyrəmək üçün User sinifində olan bütün dəyişənlər ilə əlaqə saxlanılacaq. PHP, belə məlumatlar üçün statik xüsusiyyətləri və metodları təklif edir.

Nümunə 5.18-də sinif nümayəndələrinin statik elanı göstərilmişdir və bu zaman sinifin nüsxələrinin yaradılmasına ehtiyac yoxdur. Statik elan edilmiş xüsusiyyət bilavasitə sinifin nüsxəsində əlçatan deyil. Lakin statik metoddan əlçatan ola bilər.

Nümunə 5.23-də statik xüsusiyyətə və açıq metoda malik Test adında sinif təyin edilir.

Nümunə 5.23. Statik xüsusiyyətli sinifin təyini

```
<?php
  $temp = new Test();
  echo "Test A: ". Test: :$static_property. "<br>";
  echo "Test B: ". $temp->get_sp(). "<br>";
  echo "Test C: ". $temp->static_property. "<br>";
  class Test
  {
    static $static_property = "Bu statik xüsusiyyətdir";
    function get_sp()
    {
     return self: :$static_property;
    }
  }
?>
```

Kod icra edildikdə, aşağıdakı informasiya ekranlaşdırılacaq:

```
Test A: Bu statik xüsusiyyətdir
Test B-: Bu statik xüsusiyyətdir

Notice: Undefined property: Test: :$static_property
Test C:
```

Bu nümunədə göstərilmişdir ki, Test A-da cüt iki nöqtə operatorundan istifadə edərək sinifdən birbaşa olaraq `$static_property` xüsusiyyətinə istinad etmək olar. Test B həmçinin Test sinifindən yaradılmış `$temp` obyektinin `get_sp` metodunun çağırışı yolu ilə onun qiymətini ala bilər. Amma Test C uğursuzluğa düçar olur, çünki `$static_property`-in statik xüsusiyyəti `$temp` obyektinə əlçatmazdır.

Nəzərə alın ki, `get_sp` metodu `self` açar sözündən istifadə edərək `$static_ property` xüsusiyyətinə giriş edir. Məhz bu yolla sinifin daxilindəki statik xüsusiyyətə və ya sabitə bilavasitə giriş əldə etmək mümkündür.

Varislik

sinif yaradılan anda, ondan yarımsinifi almaq mümkün olduğu kimi. Bu, vaxta çox qənaət edəcək: kodun son dərəcə dəqiq köçürülməsi əvəzinə, ona oxşar sinif götürmək mümkün olacaq, yaratmaq, onu yarımsinifə yaymaq və sadəcə o yerlərə dəyişikliklər etmək lazımdır, hansılar ki, tipik xüsusiyyətlərə malik olacaqlar. Bu, `extends` təlimatından istifadənin hesabına əldə edilir.

Nümunə 5.24-də Subscriber sinifi extends təlimatından istifadə yolu ilə User yarımsinifiylə elan edilir.

Nümunə 5.24. Varislik və sinifin yayılması

```
<?php
$object = new Subscriber;
$object->name = "Fred";
$object->password = "pword";
$object->phone = "012 345 6789";
$object->email = "fred@bloggs.com";
$object->display();
class User
{
  public $name, $password;
  function save_user()
  {
```

```
        echo "Burada istifadəçinin məlumatlarını ehtiva edən kod
yerləşir";
      }
   }
   class Subscriber extends User
   {
     public $phone, $email;
     function display()
     {
       echo "Name: ". $this->name. "<br>";
       echo "Pass: ". $this->password. "<br>";
       echo "Phone: ". $this->phone. "<br>";
       echo "Email: ". $this->email;
     }
   }
   ?>
```

`User` sinifində ilk olaraq iki xüsusiyyət — `$name` və `$password`, həmçinin məlumat bazasında cari istifadəçinin məlumatlarının saxlanılması üçün metod mövcuddur. `Subscriber` yarımsinifi əlavə edilməsi hesabına bu sinif iki xüsusiyyət — `$phone` və `$email` genişlənilir, həmçinin `$this` dəyişənindən istifadə edərək cari obyektin xüsusiyyətlərini əks etdirən metod əlavə edilir. Bu dəyişən giriş etdiyi obyektin cari qiymətinə istinad edir. Bu kod aşağıdakı informasiyanı ekranlaşdıracaq:

```
Name: Fred
Pass: pword
Phone: 012 345 6789
Email: fred@bloggs.com
```

parent təlimatı

Yarımsinifdə bu adda metod yaradılırkən, artıq bu yarımsinif valideyn sinifində iştirak edir, valideyn sinifindəki təlimatlar bu yarımsinifə köçürülür.

Bəzən belə davranış sizin üçün ürəkaçan olmaya bilər və sizə valideyn metoduna giriş əldə etmək lazım gələ bilər. Bunun üçün

nümunə 5.25-də göstərildiyi kimi parent təlimatından istifadə etmək olar.

Nümunə 5.25. Metodun köçürülməsi və parent təlimatından istifadə

```
<?php
  $object = new Son;
  $object->test();
  $object->test2();
  class Dad
  {
    function test()
    {
      echo "[Class Dad] - Mən sənin atanam<br>";
    }
  }
  class Son extends Dad
  {
    function test()
    {
      echo "[Class Son] Mən Lukam<br>";
    }
    function test2()
    {
      parent: :test();
    }
  }
?>
```

Bu kod `Dad` (Ata) adında sinif, sonra isə `Son` (Oğul) adında yarımsinif yaradır, hansı ki, xüsusiyyətlərinin və valideyn sinifi metodlarının varisi olur və sonra test metoduna köçürülür. Buna görə də, kodun ikinci sətirində test metodu çağırılarkən, yeni metod yerinə yetirilir. O variantda test-in köçürdülmüş metodunun icrasının (ifasının) tək üsulu, hansında ki, o `Dad` sinifində mövcuddur, `parent` təlimatından istifadəylə nəticələnir (ibarətdir), göstərildiyi kimi `Son` sinifinin `test2`-i funksiyada (rolda). Bu kod aşağıdakı informasiyanı ekranlaşdıracaq:

```
[Class Son] Mən Lukam
[Class Dad] Mən sənin atanam
```

Əgər metodun çağırışını cari sinifdən təmin etmək lazımdırsa, self açar sözündən istifadə etmək olar:

```
self::method();
```

Yarımsinif konstruktorları

sinifin yayılması və şəxsi konstruktorun elanı Vaxtı yarımsinifin Konstruktorları siz bilməlidir ki, PHP valideyn sinifinin metod-konstruktoru avtomatik səbəb olmağa (çağırmağa) başlamayacaq. Ki, inisializasiyanın bütün kodunun icrası (ifası) təmin olunsun, yarımsinif, göstərildiyi kimi nümunə 5.26-da, valideyn konstruktorları səbəb olmaq (çağırmaq) həmişə məcburdur.

Nümunə 5.26. Valideyn sinifinin konstruktorunun çağırışı

```
<?php
  $object = new Tiger();
  echo "pələnglərdə var.. .<br>";
  echo "Xəz: ". $object->fur. "<br>";
  echo "Zolaqlar: ". $object->stripes;
  class Wildcat
  {
    public $fur; //Vəhşi pişiklərdə xəz var
    function __construct()
    {
      $this->fur = "TRUE";
    }
  }
  class Tiger extends Wildcat
  {
    public $stripes; //Pələnglərdə zolaqlar var
    function __construct()
    {
      parent: :__construct(); //İlk valideyn konstruktorunun çağırışı
      $this->stripes = "TRUE";
    }
```

```
  }
?>
```

Bu nümunədə varisliyin adi üstünlüklərindən istifadə olunur. `Wildcat` (Vəhşi pişik) sinifində çoxqat istifadəsi nəzərdə tutulan `$fur` (xəz) xüsusiyyəti yaradılır. Buna görə də biz `$fur` xüsusiyyətini varis olan `Tiger` sinifini yaradırıq və əlavə olaraq daha bir — `$stripes` (zolaqlar) xüsusiyyətini yaradırıq. Hər iki konstruktorun çağırışını yoxlamaq üçün, proqram aşağıdakı informasiyanı ekranlaşdıracaq:

```
Pələnglərdə var...
Xəz: TRUE
Zolaqlar: TRUE
```

Final Metodları

Zəruri hallarda, yarımsinifə supersinif metodunun köçürülməsinin qarşısını almaq üçün `final` açar sözündən istifadə etmək olar. Açar sözündən istifadə, nümunə 5.27-də göstərilmişdir.

Nümunə 5.27. Final metodunun yaradılması

```
<?php
  class User
  {
   final function copyright()
   {
     echo "Bu sinif Co Smit tərəfindən yaradılmışdı ";
   }
  }
?>
```

Bu fəsilin məzmununu mənimsədiyiniz halda, PHP-nin nəyə qadir olduğunu təsəvvür edəcəksiniz. Siz, heç bir əmək tələb etməyən xüsusi funksiyalardan yararlana və zəruri olduqda şəxsi obyekt yönümlü kodunuzu yarada biləcəksiniz. 6-cı fəsildə biz PHP-nin ilkin tədqiqatını bitirəcəyik və massivlərlə işləmək qaydası ilə tanış olacağıq.

Suallar

5.1 Funksiyalardan istifadənin hansı əsas üstünlük var?

5.2 Funksiya nə qədər qiymət qaytara bilər?

5.3 Dəyişənə adı və istinad üzrə girişin fərqi nədir?

5.4 PHP-də "görünmə sahəsi" termini nəyi bildirir?

5.5 Koda digər bir PHP faylını necə daxil etmək olar?

5.6 Obyekt funksiyadan nə ilə fərqlənir?

5.7 PHP-də yeni obyektlər necə yaradılır?

5.8 Mövcud sinifdən yarımsinifin yaradılması üçün hansı sintaksisdən istifadə olunur?

5.9 Obyektin yaradılması zamanı kodun inisializasiya edən hissəsini doğurmaq (çağırmaq) olar?

5.10 Nəyə görə sinifin daxilində olan xüsusiyyətləri açıq-aydın elan etmək daha məqsədə uyğundur?

Bu sualların cavabları Əlavələrin A başlığında yerləşdirilmiş, "5-ci fəsilin suallarının cavablarının" bölməsində tapmaq olar.

5

Massivlər

Biz artıq 3-cü fəsildə massivlərlə qısa tanışlıq etdik və onların imkanları haqqında ilkin təsəvvürü formalaşdırdıq. Bu fəsildə, massivlərlə işləmək qaydalarının böyük arsenalı nümayiş etdiriləcək. Bəzi massivlərlə işləyərkən, sizdə ciddi tipləşdirmə dillər ilə iş təcrübəsi yaranacaq, məsələn C, öz sadəliyiylə və gözəlliyilə sizi təəccübləndirə bilər. Bundan başqa, bu tip proqramlaşdırma dilləri, mürəkkəb struktura malik məlumatlarla problemsiz və sürətli işləmək imkanına malikdir.

Massivlərə əsas yanaşmalar

Əvvəlki bölmələrdə massivlər, bir-birinə yapışdırılmış kibrit qutuları qrupu şəklində göstərilmişdi. Massivləri həmçinin, sap üzərində düzülmüş muncuqlar şəklində də təsəvvür etmək olar ki, burada muncuq dənələri ədəd, sətir tipində, həmçinin massivdən ibarət ola bilən dəyişənləri ifadə edir. Massivlər muncuqların düzüldüyü sapa oxşayır, çünki hər element şəxsi yerə malikdir və hər elementin (birincidən və sonuncudan başqa) hər iki tərəfində digər elementlər yerləşir.

Massivlərin hissələrinə istinad, ədəd indeksləri üzrə aparılır, lakin digər — hərf-rəqəm identifikatorları vasitəsilə də təyin etməyə icazə verilir. İnteqrasiya edilmiş funksiyaları xüsusi dövrdə istifadə edərək, hissələri çeşidləmək, əlavə etmək/silmək və onların hər birinin emalı üçün xüsusi olaraq elementlər seçmək olar. Başqa massivlərin daxilində bir və ya bir neçə massivin yerləşdirilməsi hesabına istənilən ölçülü massiv yaratmaq olar.

Massivlərin ədəd indeksləşdirməsi

Fərz edək ki, sizə şirkət üçün ofis məhsullarının çatdırılması üzrə sadə sayt yaratmaq tapşırılıb. Bu halda da siz alıcıya təqdim edilən müxtəlif kağız növlərini sayta daxil etməlisiniz. Bu müxtəlifliyi massivlərdən istifadə edərək kateqoriya şəklində formalaşdırıb, idarə etmək mümkündür. Belə yanaşmanın reallaşdırılmasının ən sadə üsulu nümunə 6.1-də göstərilmişdir.

Nümunə 6.1. Massivə elementlərin əlavə edilməsi

```
<?php
  $paper [] = "Copier";
  $paper [] = "Inkjet";
  $paper [] = "Laser";
  $paper [] = "Photo";
  print_r ($paper);
?>
```

Hər mənimsəmə vaxtı bu nümunədə boş yer sonun saxlanılması üçün qiymətin (mənanın) `$paper`-i massivinə birincini istifadə olunur, və (amma) daxili göstəricinin (siyahının) PHP-ında mövcud olanın qiyməti (mənası) vahidə böyüyür, qiymətin (mənanın) növbəti yerləşdirməsi üçün hazır boş yeri göstərmək üçün. Artıq bizə, print_r (hansı ki, dəyişən, massiv və ya obyektin tərkibini ekrana çıxarır) funksiyası bizə məlumdur və nümunədə bu funksiya, massivin doldurulmasının düzgünlüyünün yoxlanılması üçün tətbiq edilir. Funksiyanın işinin nəticəsində aşağıdakı növdə məlumat əldə olunacaq:

```
Array
(
  [0] => Copier
  [1] => Inkjet
  [2] => Laser
  [3] => Photo
)
```

Əvvəlki kod, nümunə 6.2-də göstərildiyi kimi yazılmış ola bilər. Bu zaman massivdə hər bir element üçün dəqiq yer göstərilir. Amma, gördüyünüz kimi, belə yanaşma artıq simvollar dəstini tələb edir və nəticədə əgər massivə mallar əlavə etmək və ya hər hansı malı silmək lazım olduqda, kodunuzla işləmək çətinləşir. Buna görə də, əgər massivdəki elementlərin yerləşməsi hər hansı önəm kəsb etmirsə, yerləşdirməni PHP-nin ixtiyarına verməyiniz məsləhətdir.

Nümunə 6.2. Massivə konkret göstərişlə yerləşdirilmiş elementlərin əlavə edilməsi

```
<?php
  $paper [0] = "Copier";
  $paper [1] = "Inkjet";
  $paper [2] = "Laser";
  $paper [3] = "Photo";
  print_r ($paper);
?>
```

Bu nümunə də, əvvəlki koddakı informasiyanı ekranlaşdıracaq, amma hazırlanan saytda siz çətin ki, print_r funksiyasından istifadə edəcəksiniz, buna görə də dövrün köməyilə saytda təklif edilən müxtəlif kağız tipləri haqqında məlumatların necə çap edilməsi nümunə 6.3-də göstərilmişdir, necə .

Nümunə 6.3. Massivin və massivdən çıxarılmış elementlərin göstərilməsi

```
<?php
  $paper [] = "Copier";
  $paper [] = "Inkjet";
  $paper [] = "Laser";
  $paper [] = "Photo";
  for ($j = 0; $j < 4; ++$j)
  echo "$j: $paper [$j] <br>";
?>
```

Bu nümunə aşağıdakı informasiyanı ekranlaşdıracaq:

```
0: Copier
1: Inkjet
2: Laser
3: Photo
```

Beləliklə, siz massivə elementlərin əlavə edilməsi və onlara istinad üsullarından birinin iki üsulu gördünüz, amma PHP təklif edir və çox başqa üsul, hansılarda ki, mən gələcəkdə qısa dayanacağam. Əvvəlcə digər massiv tipinə baxacağıq.

Assosiativ massivlər

Elementlərin indekslər üzrə yerləşdirilməsi və massivdən tələb olunan elementin çıxarışı üçün sadəcə elementin indeksini yazmaq çox rahat əməliyyatdır. Amma bir məsələ də var ki, məsələn hansı malın hansı nömrəyə (indeksə) yadda saxlamaq, ən azından massivə nəzər yetirmək lazımdır və massiv elementlərinin sayı çox olduqda

bu prosesin mürəkkəbliyini təsəvvür etmək elədə çətin deyil. Bax, belə hallarda assosiativ massivlərə müraciət etmək ən doğru qərardır.

Bu massivlərdən istifadə edərkən, massivin elementlərinə nömrələr üzrə deyil, adlar üzrə istinad edilməsinə icazə verir. Nümunə 6.4-də əvvəlki kodun genişləndirilmiş versiyası göstərilmişdir. Nümunədə massivin elementinin qiyməti ilə adının eyniləşdirilməsi, massivin hər bir elementinin daha uzun və informativ sətir qiyməti ilə təqdim edilməsi göstərilmişdir.

Nümunə 6.4. Assosiativ massivə elementlərin əlavə edilməsi və bu elementlərin çıxarışı

```
<?php
  $paper ['copier'] = "Copier & Multipurpose";
  $paper ['inkjet'] = "Inkjet Printer";
  $paper ['laser'] = "Laser Printer";
  $paper ['photo'] = "Photographic Paper";
  echo $paper ['laser'];
?>
```

İndi baxın ədədlərin (heç bir faydalı məlumatı ehtiva etməyən, massivdə elementin mövqeyindən başqa) yerinə hər elementin unikal ad mövcuddur ki, istinad zamanı bu addan istifadə edilir. Nümunədə gördüyünüz kimi echo təlimatında qeyd edilən açar sözü vasitəsilə, `Laser Printer` ekranlaşdırılacaq. Adlar (copier, inkjet və t. d.) indeks və ya açar, onlara mənimsənmiş elementlər isə (məsələn, Laser Printer) — qiymətlər (mənalar) adlanır.

PHP-nin olduqca güclü bu xüsusiyyətindən XML və HTML kodundan informasiyanın çıxarışı zamanı tez-tez tətbiq edilir. Məsələn, axtarış sistemində istifadə edilən HTML-parser, veb səhifənin bütün elementlərini assosiativ massivə yerləşdirə bilər. Bu zaman massivdəki adlar səhifə strukturunu əks etdirir:

```
$html ['title'] = "Mənim veb səhifəm";
$html ['body'] = "... veb səhifənin bədəni...";
```

Tamamilə ehtimal etmək olar ki, proqram istinadın səhifəsində bütün tapılmış nəticələri, başlıqları, yarımbaşlıqları qruplaşdıracaq və onları

başqa bir massivə yerləşdirəcək. Assosiativlərdən istifadə zamanı, və (amma) deyil say massivləri kod, bütün bu elementlərə istinad edən, daha sadə yaradacaq və qaydaya salacaq.

array açar sözündən istifadə etməklə mənimsətmə

Biz artıq bilirik ki, massivə qiymətlər ardıcıl əlavə edilmə yolu ilə massivin elementləri şəklində mənimsədilir. Açar sözlər, ədəd identifikatoru və ya susmaya görə indeksləşmədən istifadə etməyinizdən asılı olmayaraq, bu çox vaxt aparan prosesdir. Nümunə 6.5-də qiymətlərin bu üsulla mənimsədildiyi hər iki — həm ədəd, həm də assosiativ massiv göstərilmişdir.

Nümunə 6.5. Array açar sözündən istifadə etməklə massivə elementlərin əlavə edilməsi

```
<?php
  $p1 = array (" Copier", "Inkjet", "Laser", "Photo");
  echo "p1 massivinin elementi: ". $p1 [2]. "<br>";
  $p2 = array ('copier' => "Copier & Multipurpose",
               'inkjet' => "Inkjet Printer",
               'laser' => "Laser Printer",
               'photo' => "Photographic Paper");
  echo "p2 massivinin Elementi: ". $p2 ['inkjet']. "<br>";
?>
```

Bu kod fraqmentinin birinci hissəsində malın köhnə, qısaldılmış təsviri $p1 massivinə mənimsədilir. Burada dörd element istifadə olunur ki, onlar 0 - 3 aralığındakı mövqeləri tutur. echo təlimatı aşağıdakı mətni ekranlaşdırır:

```
p1 massivinin elementi: Laser
```

Kodun ikinci hissəsində assosiativ identifikatorlar və onları müşayiət edən malların uzun təsvirləri $p2 massivinə mənimsədilir. Bunun üçün *indeks=> qiymət* formatı tətbiq edilir. Operatorun tətbiqi => görünür mənimsəmənin sadə operatorundan istifadəyə =, ondan başqa ki, qiymət (məna) indeksə mənimsənilir, və (amma) dəyişən

deyil. Bundan sonra indeks ona digər qiymət mənimsədilməyənədək bu qiymətlə qırılmaz əlaqəyə malik olur. Buna görə də, echo komandası aşağıdakı mətni ekranlaşdırır:

```
p2 massivinin elementi: Inkjet Printer
```

Onda, müəyyən etmək olar ki, $p1 və $p2 massivləri müxtəlif tiplərə aiddir. Əmin olmaq olar, əgər koda qeyri-müəyyən indeksin səhvinə və ya qeyri-müəyyən yerdəyişmənin səhvinə səbəb olan (çağıran) iki növbəti komanda yerləşdirməksə bir halda ki, (çünki,) massivlərdən hər biri üçün uyğun olmayan identifikator istifadə olunur:

```
echo $p1 ['inkjet']; // Qeyri-müəyyən indeksi
echo $p2 ['3'] ;     // Qeyri-müəyyən yerdəyişmə
```

foreach...as dövrü

PHP-ni hazırlayanlar daimi, bu dilin istifadəsini kifayət qədər sadə etməyə çalışırlar. Bunun üçün də onlar, dövrlərin təşkilində mövcud strukturlar ilə kifayətlənməyib, massivlər üçün xüsusi olaraq nəzərdə tutulmuş daha bir strukturu — `foreach...as` dövrünü əlavə etdilər. Bu dövrdən istifadə edərək, massivin bütün elementlərini seçmək və onlarla növbəli ilə hər hansı əməliyyatları yerinə yetirmək olar. Proses birinci elementdən başlanır və son elementdə bitir, buna görə də, massivin nə qədər element ehtiva etdiyini bilməyiniz vacib deyil. Nümunə 6.6, foreach...as dövründən istifadə etməklə nümunə 6.3 kodunun köçürülməsi göstərilmişdir.

Nümunə 6.6. foreach...as dövründən istifadəylə ədəd massivinin elementlərinin ardıcıl izafəsi

```
<?php
  $paper = array (" Copier", "Inkjet", "Laser", "Photo");
  $j = 0;
  foreach ($paper as $item)
  {
    echo "$j: $item<br>";
    ++$j;
  }
```

```
?>
```

PHP, `foreach` təlimatı ilə qarşılaşdığı zaman, ilk olaraq, massivin birinci elementini çıxardır və çıxardılmış elementi `as` açar sözünün qarşısında göstərilmiş dəyişənə yerləşdirir. Quraşdırılmadan asılı olaraq, foreach təlimatının hər qayıdışı anında, massivin növbəti elementinin qiyməti də ardıcıllıqla bu dəyişənə yerləşdirilir. Uyğun olaraq, indiki halda da $paper massivində saxlanılan dörd qiymət növbə ilə $item dəyişəninə mənimsədilir. Bütün qiymətlərdən istifadə edildikdən sonra, dövrün icrası bitir. Bu kodun nəticəsi, nümunə 6.3-dəki kodun nəticəsi ilə eynidir.

İndi isə foreach-in, assosiativ massivlə necə işlədiyinə baxacağıq. Nümunə 6.7-də nümunə 6.5-in ikinci hissəsi köçürdülmüşdür.

Nümunə 6.7. Foreach...as dövründən istifadə etməklə assosiativ massivin elementlərinin ardıcıl izafəsi

```
<?php
  $paper = array ('copier' => "Copier & Multipurpose",
                  'inkjet' => "Inkjet Printer",
                  'laser' => "Laser Printer",
                  'photo' => "Photographic Paper");
  foreach ($paper as $item => $description)
  echo "$item: $description<br>";
?>
```

Qeyd etdiyimiz kimi, assosiativ massivlərdə ədəd indeksləşməsi tələb olunmur, buna görə də bu nümunədə `$j` dəyişənindən istifadə olunmur. Bunun yerinə `$paper` massivinin hər bir elementi təqdim edilmiş `$item` və `$description` dəyişənlərinə *"açar – qiymət"* cütlüyü şəklində daxil edilir və bu cütlük aşağıdakı qaydada ekranlaşdırılır:

```
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

`foreach...as` sintaksisinə alternativ olaraq each və bu funksiyaya uyğun olan `list` funksiyasından istifadə etmək olar (nümunə 6.8).

Nümunə 6.8. each və list funksiyalarının köməyilə assosiativ massivin elementlərinin ardıcıl izafəsi

```
<?php
  $paper = array ('copier' => "Copier & Multipurpose",
                  'inkjet' => "Inkjet Printer",
                  'laser' => "Laser Printer",
                  'photo' => "Photographic Paper");
  while (list ($item, $description) = each ($paper))
  echo "$item: $description<br>";
?>
```

Bu nümunədə `each` funksiyası `FALSE` qiymətini alanadək işin icrasını davam etdirməlidir və buna görə də `while` dövrü qurulur. `each` funksiyasının davranışı `foreach`-in davranışına bənziyir: o `$paper` massivindən *"açar — qiymət"* cütlüyünü ehtiva edən massivi alır və sonra ilk massivdə növbəti cütlüyə inteqrasiya edilmiş göstərici (siyahı) şəklində formalaşdırır. `each` funksiyası `FALSE` qiymətini aldığı anda proses bitir.

`list` funksiyası, massivi (indiki halda, each funksiyası ilə alınmış *"açar — qiymət"* cütlüyünü) arqumentlər şəklində qəbul edir və sonra massivin qiymətlərini yumru mötərizələrin daxilində göstərilmiş dəyişənlərə mənimsədir.

`list` funksiyasının iş prinsipini daha yaxşı nümunə 6.9-da anlamaq olar. Nümunədə massiv iki sətir — Alice və Bob sətiri kimi yaradılır və sonra da `list` funksiyasına ötürülür. Ötürülən sətirlər isə müvafiq olaraq `$a` və `$b` dəyişənlərinə mənimsədilir.

Nümunə 6.9. List funksiyasından istifadə qaydası

```
<?php
  list ($a, $b) = array ('Alice', 'Bob');
  echo "a=$a b=$b";
? >
```

Bu kod aşağıdakı mətni ekranlaşdıracaq:

```
a=Alice
b=Bob
```

Beləliklə, massivin elementlərinin izafəsi üçün müxtəlif yanaşmaları tətbiq etmək olar. Bunun üçün `foreach...as` konstruksiyasından və ya `each` funksiyasından istifadə etmək olar.

Çoxölçülü massivlər

PHP-də massiv sintaksisinin sadə konstruktiv xüsusiyyəti, çoxölçülü massivləri bir ölçüdə yaratmağa imkan verir.

Faktiki olaraq ixtiyari ölçülü massiv yaratmaq olar (hərçənd ki, proqramlarda nadir hallarda üç ölçüdən yuxarı massivlərə rast gəlinir). Bu xüsusiyyət bütöv massivi başqa massivin tərkibinə daxil etmək, həmçinin daxil edilən edilən massivin daxilində massiv yerləşdirmək olar. Beləliklə, təkrar-təkrar əməliyyatlar etmək olar, necə birələr haqqında köhnə şeirdə, hansılar ki, pomenşenin başqa birələri dişləyirlər, və (amma) onlar, öz növbəsində, öz birələri dişləyirlər, və belə sonsuza qədər.

Gəlin, bunun necə işlədiyinə baxaq. Əvvəlki nümunəki assosiativ massivi götürərək, onu genişləndirək (nümunə 6.10-u).

Nümunə 6.10. Çoxölçülü assosiativ massivin yaradılması

```
<?php
  $products = array ('paper' => array ('copier' => "Copier &
                      Multipurpose", 'inkjet' => "Inkjet Printer",
                      'laser' => "Laser Printer", 'photo' =>
                      "Photographic Paper"), 'pens' => array ('ball' =>
                      "Ball Point", 'hilite' => "Highlighters", 'marker'
                      => "Markers"), 'misc' => array ('tape' => "Sticky
                      Tape", 'glue' => "Adhesives", 'clips' =>
                      "Paperclips"));
  echo "<pre>";
  foreach ($products as $section => $items)
  foreach ($items as $key => $value)
  echo "$section:\t$keyt ($value) <br>";
  echo "</pre>";
```

```
?>
```

Genişlənmə anlayışını sadələşdirmək üçün, mən elementlərin hissə adını dəyişdirdim. Məsələn, əvvəlki $paper massivi daha böyük massivin yarımfəsili oldu, əsas massiv isə, indi $products adlanır. Bu massivdə üç element mövcuddur: kağız — paper, qələmlər (qulplar) — pens və müxtəlif mallardır — misc və onlardan hər biri özündə "açar — qiymət" cütlüklərdən ibarət olan digər bir massivi ehtiva edir. Zəruri olduqda bu altmassiv başqa massivləri də ehtiva edə bilər.

Məsələn, "diyircəkli qələmlər" elementi — ball — bir çox internet mağazasında olan bu malın tipi və rəngi ehtiva edilə. Amma hələ ki mən kodu iki ölçülü dərinlikdə məhdudlaşdırdım. Mənimsətmədən sonra müxtəlif qiymətlərin nəticəsi üçün məlumat massivlərində foreach...as dövründən istifadə etdim. Xarici dövr yuxarı səviyyəli massivdən əsas bölmələri çıxardır. Daxili dövr isə "açar — qiymət" cütlüyünün hər bölməsinin kateqoriyalardırılması üçün çıxardır. Massivin bütün səviyyələrinin ("açar — qiymət" cütlüyünün timsalında) eyni qaydada işlədiyini nəzərə alaraq, heç bir xüsusi əmək tələb olunmadan, istənilən səviyyədə istənilən elementə giriş kodunu yaratmaq olar. Echo təlimatında tabulyasiya nişanını bildirən t PHP idarəedici simvolundan istifadə olunur. Hərçənd ki, tabulyasiya nişanları brauzerlər üçün bir qayda olaraq, heç bir məna kəsb etmir, ona görə də mən nişanı <pre>.. .</pre> teqləri daxilində qeyd etdim ki, brauzerə ilkin formatın və fiksə enin saxlanılmasıyla mətni format etmək və tabulyasiyanın və sətirin tərcümələrinin (köçürmələrinin) nişanları kimi əks etdirilməyən simvollara etinasızlıq göstərməmək əmr verirlər.

Bu kodun icrası nəticəsində aşağıdakı növdə mətn ekranlaşdırılacaq:

```
paper: copier (Copier & Multipurpose)
paper: inkjet (Inkjet Printer)
paper: laser (Laser Printer)
paper: photo (Photographic Paper)
pens: ball (Ball Point)
```

```
pens: hilite (Highlighters)
pens: marker (Markers)
misc: tape (Sticky Tape)
misc: glue (Adhesives)
misc: clips (Paperclips)
```

Massivin konkret elementinə Bilavasitə giriş etmək üçün kvadrat mötərizələrdən istifadə etmək lazımdır:

```
echo $products ['misc'] ['glue'];
```

Bu kod Adhesives qiymətini çıxarır. Həmçinin, elementlərinə bilavasitə girişi hərf-rəqəm identifikatorları üzrə deyil, indekslər üzrə mümkün olan çoxölçülü ədəd massivi yaratmaq olar.

Nümunə 6.11-də fiqurları düzülmüş şahmat lövhəsi yaradılır. Nümunə 6.11. Çoxölçülü ədəd massivinin yaradılması

```
<?php
  $chessboard = array (array ('r', 'n', 'b', 'q', 'k', 'b', 'n',
                        'r'), array ('p', 'p', 'p', 'p', 'p', 'p',
                        'p', 'p'), array (' ', ' ', ' ', ' ', ' ', '
                        ', ' ', ' '), array (' ', ' ', ' ', ' ', ' ',
                        ' ', ' ', ' '), array (' ', ' ', ' ', ' ', '
                        ', ' ', ' ', ' '), array (' ', ' ', ' ', ' ',
                        ' ', ' ', ' ', ' '), array ('P', 'P', 'P',
                        'P', 'P', 'P', 'P', 'P'), array ('R', 'N',
                        'B', 'Q', 'K', 'B', 'N', 'R'));
  echo "<pre>";
  foreach ($chessboard as $row)
  {
  foreach ($row as $piece)
  echo "$piece "; echo "<br>";
  } echo "</pre>";
?>
```

Bu nümunədə kiçik hərf registri qara, böyük hərf registri isə ağ fiqurları bildirir. Aşağıdakı işarələrdən istifadə olunur: r — rook (top), n — knight (at), b — bishop (fil), k — king (şah), q — queen (vəzir) və p — pawn (piyada). Massivin ardıcıl izafəsi və onun tərkibinin nümayişi üçün iki foreach...as dövründən istifadə olunur. Xarici dövr hər bir horizontalı emal edir və onu $row dəyişəninə

yerləşdirir. Yerləşdirilən dəyişən, öz-özlüyündə də massivdir, çünki, şahmat lövhəsinin massivi hər horizontal üçün — $chessboard — altmassivindən istifadə edir. Bu dövrdə iki təlimatdan istifadə olunur, buna görə də onlar fiqurlu mötərizələrlə əhatələnmişdir. Daxili dövr isə horizontalın hər bir qəfəsini emal edir, boşluq simvolu ehtiva edən massivdən ($piece) sözügedən simvolu çıxararaq, mətni şahmat lövhəsi formasına salır.

Bu dövrdə bir təlimat ibarətdir və fiqurlu mötərizələrdən istifadəyə ehtiyac yoxdur. <pre> və </pre> teqləri çıxarılan mətnin düzgün formada yazılmasını təmin edir:

```
r n b q k b n r
p p p p p p p p
P P P P P P P P
R N B Q K B N R
```

Kvadrat mötərizələrdən İstifadə edərək, bu massivin istənilən elementinə bilavasitə giriş almaq olar:

```
echo $chessboard [7] [3];
```

Bu zaman echo təlimatı, aşağı horizontalın səkkizinci, dördüncü şaquli elementin qiymətini, yəni böyük hərf registrində Q hərfini ekranlaşdıracaq (yaddan çıxartmayın ki, massivin indeksləri vahiddən deyil sıfırdan başlanır.

Massiv funksiyalarından istifadə

Massivlərlə işləyərkən funksiyalardan istifadə List və each funksiyaları artıq sizə tanışdır, lakin PHP-də bu funksiyalardan sayayı massivlərlə iş üçün nəzərdə tutulmuş bir çox başqa funksiya da mövcuddur. Bu funksiyaların tam siyahısı http://tinyurl.com/phparrayfuncs ünvanında yerləşdirilmişdir. Amma bu funksiyaların bəziləri PHP proqramlaşdırmada əhəmiyyətli rola malikdir. Biz indi, sözügedən funksiyaları daha ətraflı öyrənəcəyik.

is_array

Dəyişənin massivlər sahəsindən istifadə etdiyini yoxlayır. Bu onu bildirir ki, məsələn kodda $fred adlı dəyişənin massiv və ya sətir tipində olan qiyməti ehtiva etdiyi naməlumdur və proqramın kodunuzda bunu yoxlamaq üçün is_array funksiyasından istifadə etmək olar:

```
echo (is_array ($fred))? "Bu massivdir":"Bu massiv deyil";
```

Qeyd edək ki, indiki halda $fred dəyişəninə heç bir qiymət mənimsədilməmişdir, buna görə də qeyri-müəyyən dəyişən haqqında mesaj göstəriləcək — Undefined variable.

count

Baxmayaraq ki, each funksiyası və foreach...as dövr strukturu massivin bütün tərkibinin ardıcıl izafəsini əla üsulları təşkil edir, bəzən (xüsusəndə, massivin elementlərinə birbaşa müraciət etdikdə) massivin elementlərin sayı sizə lazım olur. Massivin yuxarı səviyyəsində bütün elementlərin hesablanması üçün aşağıdakı komandadan istifadə olunur:

```
echo count($fred);
```

Əgər çoxölçülü massivdə bütün massivlər daxil olmaqla, ümumilikdə nə qədər elementin olduğunu öyrənmək istəyirsinizsə aşağıdakı təlimatdan istifadə edə bilərsiniz:

```
echo count ($fred, 1);
```

İkinci parametr vacib olmayan parametrdir və istifadə rejimini bildirir. Buraya iki cür parametr daxil edilə bilər: sıfır hesablamanı yalnız yuxarı səviyyə ilə məhdudlaşdırmaq üçün bir— altmassivlərin elementlərinin küliyyatı da daxil olmaqla rekursiv hesablanma üçün.

sort

Çeşidləmə o qədər yayılmış əməliyyatdır ki, bu əməliyyat üçün PHP-yə inteqrasiya edilmiş xüsusi funksiya mövcuddur. Funksiyanın sadə

formasından aşağıdakı qaydada istifadə etmək olar:

```
sort ($fred);
```

Bu funksiyanin digər funksiyalardan fərqi ondan ibarətdir ki, çeşidləmə əməliyyatı zamanı funksiya, bilavasitə göstərilmiş massivlə işləyir, yəni çeşidlənmiş elementlər yeni massiv şəklində formalaşmır. Çeşidləmə əməliyyatının müvəffəqiyyətli icrası zamanı funksiya TRUE, əməliyyat zamanı xəta baş verərkən isə FALSE qaytarır. Bu funksiya həmçinin bir neçə xüsusi olaraq nəzərdə tutulmuş bayraq dəstəkləyir. Bunlardan ən əsası aşağıdakılardır, hansı ki, gələcəkdə sizə lazım ola bilər:

`sort ($fred, SORT_NUMERIC)` ədəd çeşidlənməsi;

`sort ($fred, SORT_STRING)` sətir çeşidlənməsi;

Həmçinin rsort funksiyasından istifadə edərək massivin əks çeşidlənməsi də mümkündür:

```
rsort ($fred, SORT_NUMERIC);
rsort ($fred, SORT_STRING);
```

shuffle

Elementləri təsadüfi sırada yerləşdirir. Məsələn, kart oyununda kartların (indiki halda massivin elementlərinin) təsadüfi sırada yerləşməsini istəyirsinizsə:

```
shuffle ($cards);
```

shuffle funksiyası sort funksiyası kimi, bilavasitə ona verilmiş massivlə işləyir və müvəffəqiyyətli iş anında TRUE, əks təqdirdə isə FALSE qiymətini qaytarır.

explode

Bu funksiya sətir elementlərini (söz, hərf, rəqəm, simvol və s.) massiv elementləri şəklində formalaşdırmaq bacarığına malikdir və çox faydalı funksiyadır. Nümunə 6.12-də bu funksiyanın faydalı tətbiqlərindən biri göstərilmişdir. Nümunədə, göstərilmiş cümlə,

ayırıcı parametrdə boşluq simvolu göstərildiyinə görə, sözlər şəklində qruplaşdırılaraq, müvafiq olaraq, massivin elementlərini formalaşdırır. Massivin elementlərinin indeksi, sözlərin cümlədəki yerinə müvafiq olaraq nizamlanır.

Nümunə 6.12. Ayırıcı simvolun boşluq simvolu təyin etməklə, sətirdəki sözlərin massiv şəklində formalaşdırılması

```
<?php
  $temp = explode (' ', "Bu cümlə beş sözdən ibarətdir");
  print_r ($temp);
?>
```

Bu nümunə aşağıdakı informasiyanı (brauzerdə baxış zamanı informasiya bir sətirdə əks etdiriləcək) ekranlaşdırır:

```
Array
(
  [0] => Bu
  [1] => cümlə
  [2] => beş
  [3] => sözdən
  [4] => ibarətdir
)
```

Birinci parametr, qeyd etdiyimiz kimi ayırıcı parametrdir. Bu parametrdə istənilən simvol göstərilə bilər və göstərilən simvol massivin elementlərinin formalaşdırılmasında ayırıcı rola malik olacaq. Nümunə 6.13-də həmin kod bir qədər dəyişdirilmiş formada göstərilmişdir.

Nümunə 6.13. *** simvollar ilə bölünmüş sözlərin, sətirdən massivə çıxardılması

```
<?php
$temp = explode ('***', "Bu***sözlər***ulduzlarla*** bölünmüşdür");
print_r ($temp);
?>
```

nümunə 6.13 Kodun icrası nəticəsində aşağıdakı informasiya alınır:

```
Array
```

```
(
  [0] => Bu
  [1] => sözlər
  [2] => ulduzlarla
  [3] => bölünmüşdür
)
```

extract

Bəzən "açar — qiymət" cütlüklərini massivdən PHP dəyişənlərinə çevirmək daha rahatdır.

Belə hadisələrdən biri — forma vasitəsilə PHP ssenarisinə göndərilmiş $_GET və ya $_POST dəyişənlərinin emalıdır. Forma İnternet vasitəsilə ötürələrkən, veb-server dəyişənləri açır və onları PHP ssenarisi üçün nəzərdə tutulmuş qlobal massivə yerləşdirir. Əgər dəyişənlər GET metodu ilə göndərilmişdirsə, onlar $_GET metodunun assosiativ massivinə, POST metodu ilə göndərilmişdirsə, onlar $_POST metodunun assosiativ massivinə yerləşdiriləcək. Əlbəttə ki, bu fəsildə göstərilmiş üsullardan istifadə edərək, assosiativ massivlərin bütün elementlərini seçmək olar. Amma bəzən göndərilmiş qiymətləri dəyişənlərdə yalnız sonrakı istifadə üçün saxlamaq lazım gəlir. Belə hallarda PHP, bunu avtomatik rejimdə aşağıdakı sintaksis vasitəsilə edə bilər:

```
extract ($_GET);
```

Beləliklə, məsələn, əgər q sorğusunun sətri parametri onunla bağlı Hi there qiymətilə PHP ssenarisinə göndərilmişsə, $q adlı yeni dəyişən yaradılacaq və bu qiymət yaradılan dəyişənə mənimsədiləcək. Amma göstərilmiş yanaşma ilə biraz ehtiyatlı olmaq lazımdır, çünki, əgər çıxardılan dəyişənlər artıq elan edilmiş ola bilər və onda mövcud dəyişənlər köçürdüləcək. Belə məqamlardan sığortalanmaq üçün, bu funksiyaya əlçatan olan bir çox əlavə parametrlərdən istifadə etmək olar:

```
extract ($_GET, EXTR_PREFIX_ALL, 'fromget');
```

Bu halda yaradılan bütün yeni dəyişənlərin adları sətir xəttinin müşayiəti ilə göstərilmiş prefiks ilə başlanacaq, buna görə də $q $fromget_q-ə çevriləcək. Mən $_GET və ya $_POST istifadəsi zamanı israrla açarların tətbiqini məsləhət görürəm, belə olan halda istifadəçi əməliyyata tam nəzarət edir. Çünki, kiber-cinayətkarlar sıx istifadə edilən dəyişən adlarından istifadə edərək, sizə xüsusi olaraq seçilmiş açarlar göndərə bilər və bu halda saytınız təhlükəsizliyi zərbə altında qala bilər.

compact

Bəzən, xüsusilə extract funksiyasının tətbiqinin ziddiyyətli olması halında, dəyişənlər və onların qiymətlərindən ibarət massiv yaratmaq üçün compact funksiyasından istifadə etmək lazımdır. Bu funksiyanın tətbiqi nümunə 6.14-də göstərilmişdir.

Nümunə 6.14. Compact funksiyasından istifadə qaydası

```
<?php
$fname = "Doctor";
$sname = "Who";
$planet = "Gallifrey";
$system = "Gridlock";
$constellation = "Kasterborous";
$contact = compact ('fname', 'sname', 'planet', 'system', 'constellation');
print_r ($contact);
?>
```

nümunə 6.14-dən kodun icrası Nəticəsində aşağıdakı informasiya göstəriləcək:

```
Array
(
  [fname] => Doctor
  [sname] => Who
  [planet] => Gallifrey
  [system] => Gridlock
```

```
    [constellation] => Kasterborous
)
```

Nəzərə alın ki, compact funksiyalarının tətbiqi zamanı dəyişənlərin adlarında $ simvolundan istifadə etmək olmaz və bu adlar dırnaqarasında yazılmalıdır. Bunun səbəbi odur ki, compact funksiyası dəyişənləri ad siyahısında axtarır. Bu funksiyandan həmçinin bir neçə dəyişənin qiyməti ilə birlikdə eyni anda nəzərdən keçirdilməsində və nəzərdən keçirilən dəyişənlərin qiymətlərindən ibarət massiv yaradılmasında da istifadə etmək olarn (nümunə 6.15).

Nümunə 6.15. compact funksiyasından fərqli istifadə

```
<?php
  $j = 23;
  $temp = "Hello";
  $address = "1 Old Street";
  $age = 61;
  print_r (compact (explode (' ', 'j temp address age')));
?>
```

nümunənin İş prinsipi, explode funksiyasından istifadə edərək sətirdəki bütün sözlərin çıxardılaraq, compact funksiyasına ötürülməsi, funksiyadan alınan massivin print_r funksiyasına ötürülməsinə əsaslanmışdır, hansı ki, sonra compact funksiyasına ötürülür, və (amma) o, öz növbəsində, print_r funksiyasının massivi qaytarır, hansı ki, nəticədə onun tərkibini göstərir. Əgər print_r funksiyasının çağırışını ehtiva edən kod sətirini kopyalasaz və harasa əlavə etsəz, onda yalnızca dəyişənlərin adlarını dəyişdirməklə, ehtiva etdikləri qiymətləri çıxartmaq olar. Bu nümunədə çıxarılan informasiya aşağıdakı növdə olacaq:

```
Array
(
    [j] => 23
    [temp] => Hello
    [address] => 1 Old Street
    [age] => 61
)
```

reset

foreach...as konstruksiyası və ya ya each funksiyalarının köməyilə massivin elementlərinin ardıcıl izafəsi təşkil olunarkən, çıxarılan növbəti massiv elementini təyin etmək üçün, PHP-nin daxilində xüsusi siyahı qurulur. Əgər proqramın kodunda massiv başlanğıcına qayıtmağa ehtiyac yaranarsa, onda reset funksiyasından istifadə etmək olar. Bu halda massiv elementlərinin növbəli emalı dayanaraq, birinci massiv elementinə qayıdılacaq. Bu funksiyadan aşağıdakı qaydada istifadə edə bilərsiniz:

```
reset($fred);         // Throw away return value
$item = reset($fred); // Keep first element of the array in $item
```

end

Massiv elementini həmçinin, PHP-nin daxili siyahısının son elementi kimi təyin etmək olar. Bunun üçün end funksiyasından istifadə edə bilərsiniz:

```
end($fred);
$item = end($fred)
```

bu fəsildə PHP-nin əsaslarının müqəddiməsi yekunlaşır. İndi, əldə etdiyiniz biliklərdən istifadə edərək, siz kifayət qədər mürəkkəb proqramlar hazırlaya bilərsiniz. Növbəti fəsildə isə ən yayılmış praktik məsələlərin həllində PHP-nin tətbiqinə baxılacaq.

Suallar

6.1 Ədəd və assosiativ massivlərin fərqi?

6.2 array açar sözündən istifadənin əsas üstünlüyü hansıdır?

6.3 foreach və each əməliyyatlarının fərqi?

6.4 Çoxölçülü massiv necə yaradılır?

6.5 Massivdə olan elementlərin miqdarını necə müəyyən etmək olar?

6.6 explode funksiyasının təyinatı nədən ibarətdir?

6.7 PHP-nin daxili siyahısını birinci massiv elementinə necə qaytarmaq olar?

6

PHP proqramlaşdırılması üzrə praktikum

Əvvəlki fəsillərdə PHP dilinin elementlərinə yer ayrılmışdır. Bu fəsil isə, tipik, lakin bununla yanaşı əhəmiyyətli praktik məsələlərin həlli prosesində proqramlaşdırma bacarıqlarının əldə edilməsi üçün nəzərdə tutulmuşdur.

Burada tamamilə aydın və yığcam kodlar vasitəsilə sətirlərin emalının ən yaxşı üsulları təqdim ediləcək ki, tarix və vaxtın təkmilləşdirilmiş təsviri, onun idarə edilməsi öyrədiləcək.

Sayta istifadəçilər tərəfindən yüklənilmiş fayllar daxil olmaqla, faylların yaradılması və dəyişikliyinin müxtəlif üsulları haqqında

artıq məlumatınız var. Bu fəsildə həmçinin HTML-ə oxşar və onun əvəzedilməsi üçün nəzərdə tutulmuş nişan dili olan XHTML-ə (məlumatların saxlanılması üçün istifadə olunan və XML sintaksisinə uyğun, məsələn RSS-axın) əsası qoyulacaq.
Məcmu halda bütün bunlar sizin tək PHP-də deyil, həm də beynəlxalq veb-standartların inkişafının praktik proqramlaşdırma sahəsində məlumat arsenalınızı genişləndirəcək.

printf funksiyasından istifadə

Artıq `print` və `echo` funksiyaları bizə tanışdır və bilirsiniz ki, bu funksiyalardan brauzerdə mətnin ekranlaşdırılması üçün istifadə olunur. Amma xüsusi formatlaşma simvollarının sətiri daxil edilməsi ilə çıxarılan məlumatların formatını idarə edən printf funksiyasının əsas üstünlüyüdür.
`printf` funksiyası gözləyir ki, hər bir formatlaşdıran simvol üçün arqument veriləcək ki, təsvir göstərilmiş formata uyğun əks olunsun. Məsələn, aşağıdakı fraqmentdə `%d` dəyişikliyinin spesifikatoru qiymətin 3 dərinlikli onluq ədəd şəklində əks olunmasını təmin edir:

```
printf ("Sizin səbətinizdə %d element mövcuddur", 3);
```

Əgər `%d`-ı `%b` ilə əvəz etsək, onda 3 qiyməti ikilik say sitemində göstəriləcək (11). Cədvəldə. 7.1 funksiyayla dəstəklənən dəyişikliyin spesifikatorları göstərilmişdir.
`printf` funksiyasında istənilən qədər spesifikatordan istifadə etmək olar. Lakin burada bir sıra şərtlər gözlənilməlidir: 1. Spesifikatorlar ilə arqumentlərin miqdarı uyğun olmalıdır; 2. hər bir spesifikator % simvolu vasitəsilə daxil edilməlidir. Buna görə də aşağıdakı kod tamamilə mümkün formaya malikdir və təklif çıxarır: *"Mənim adım Rəşaddır. Mənim 33 yaşım var, onaltılıq say sistemində isə, 21 yaşım var":*

```
printf ("Mənim adım %s. Mənim %d yaşım var, onaltılıq say sistemində
isə, %X yaşım var", 'Rəşad', 33, 33);
```

Əgər hər hansı arqumenti buraxsaq, onda sintaktis xətası yaranacaq ki, bu xətada, mötərizələrdəo sağ yumru mötərizəsinin gözlənilməz yerdə olduğu barədə bildiriləcək.
Praktik nöqteyi-nəzərdən printf funksiyasından istifadənin daha faydalı nümunəsi onluq ədədlərdən istifadə edərək HTML kodunda rəngləri təyin etməkdir. Fərz edək ki, sizə üç qiymətdən təşkil olunmuş rəng çaları lazımlıdır: qırmızı üçün — 65, yaşıl rəng üçün —

127, göy rəng üçün isə — 95, amma bu ədədlərin onaltılıq say sisteminə çevrilməlidir. Bunun üçün sadə üsul mövcuddur:

```
printf ("<span color='#%X%X%X'>Salam</span>", 65, 127, 245);
```

Rəng spesifikasiyasını diqqətlə baxın ki, onlar apostroflar (') ilə bağlanmışdır. Əvvəlcə rəng spesifikasiyasını bildirmək üçün şarp işarəsi (#) qoyulur. Sonra daxil etdiyiniz ədədlərin %X spesifikasıyası formatlaşdırması gəlir. Nəticədə bu komanda aşağıdakı mətni verəcək:

```
<span color='#417FF5'>Salam</span>
```

Adətən printf arqumentlərini dəyişənlərdən və ya ifadələrdən istifadə etməklə vermək daha rahatdır. Məsələn, əgər rənglərin təyini üçün qiymətlər üç dəyişəndə saxlanılırsa – $r, $g və $b, onda daha tünd çaları bu ifadənin köməyi ilə almaq olar:

```
printf ("<span color='#%X%X%X'>Privet</span>", $r-20, $g-20, $b-20);
```

Dəqiqliyin tənzimlənməsi

Məlumatların təqdim etməsinin (təsəvvürünün) qurması (kökləməsi) yalnız dəyişikliyin tipi göstərməmək Olar, həm də əks etdirilən nəticənin dəqiqliyi. Məsələn, valyutada məbləğlər bir qayda olaraq, iki rəqəm dəqiqliyində əks olunur. Lakin hesablanmadan sonra qiymət daha yüksək dəqiqliyə malik ola bilər (məsələn, əgər 123,42-ni 12-yə bölsək, onda 10,285 alınacaq). Belə qiymətlərin "düzgün" saxlamasını təmin etmək üçün, yəni təsviri iki rəqəm dəqiqliyi qədər təşkil etmək üçün, % simvolu ilə dəyişiklik spesifikatoru arasında ".2" sətirini yerləşdirmək olar:

```
printf ("Nəticə: $%.2f", 123.42 / 12);
```

Bu komanda aşağıdakı mətni çıxarır:

```
Nəticə: $10.29
```

Lakin mümkün idarəetmə vasitələri bununla bitmir. Məsələn, harada və nə qədər sıfır və ya boşluq göstərmək, həmçinin spesifikatordan əvvəl uyğun olan qiymətləri daxil etməklə çıxarılan mətni əlavə etmək olar. Nümunə 7.1-də beş kombinasiya göstərilmişdir.

Nümunə 7.1. Dəqiqliyin tənzimlənməsi

```
<?php
  echo "<pre>"; // Bütün boş sahələri əks etdirməyə imkan verən teq
    // 15-ə qədər işarə yeri
  printf("The result is $%15f\n", 123.42 / 12);
     // Boşluqlarla doldurulan 15 simvollu yerə malik ikidəqiqlikli
      nəticə
  printf("The result is $%015f\n", 123.42 / 12);
    // Pad to 15 spaces, 2 decimal places precision
```

```
    printf("The result is $%15.2f\n", 123.42 / 12);
      // Pad to 15 spaces, 2 decimal places precision, fill with zeros
    printf("The result is $%015.2f\n", 123.42 / 12);
       // Pad to 15 spaces, 2 decimal places precision, fill with #
        symbol
    printf("The result is $%'#15.2f\n", 123.42 / 12);
?>
```

Bu nümunə aşağıdakı mətni çıxarır:
Nəticə $ 10.285000
Nəticə bərabərdir $00000010.285000
Nəticə bərabər $ 10.29
Nəticə bərabər $000000000010.29
Nəticə bərabər $ bərabər # # # # # # # # # # 10.29 daha sadə spesifikatorun işini Təqib etmək, əgər soldan sağa onu öyrənməksə (araşdırmaqsa) (cədvəl. 7.2).
Diqqəti aşağıdakı məqamlar yönəldin.

- Ekstremal sağda indiki halda üzən nöqtəylə saya dəyişikliyi bildirən f dəyişikliyin spesifikatorunun simvoludur.
- Əgər dərhal dəyişikliyin spesifikatorundan əvvəl (qarşısında) nöqtənin və sayın uyğunluğu dəyirsə (durursa), deməlisə, çıxarılan informasiyanın dəqiqliyi bu sayla göstərilmişdir.
- dəqiqliyin spesifikatorunun olmasından Asılı olmayaraq, əgər ümumiyyətlə say spesifikator varsa, o çıxarılan informasiyanın altında seçilən (ayırılan) znako-yerlərin miqdarını təşkil edir.

Əvvəlki nümunədə bu miqdar 15 bərabərdir. Əgər çıxarılan informasiya artıq seçilən (ayırılan) znako-yerlərin miqdarına bərabərdirsə və ya onu ötür, onda bu arqument məhəl qoyulmur. ❓ ekstremaldan Sonra solda % simvolu simvol 0-ı qoymağa həll edilir (icazə verilir), hansı ki, məhəl qoyulmur, əgər seçilən (ayırılan) znako-yerlərin miqdarı göstərilməmişdirsə. Əgər bu miqdar göstərilmişdirsə, onda boşluqların yerinə əlavə sıfırlarla edilir (istehsal edilir). Əgər lazımdırsa ki, boş qalan znako-yerlər sıfırlarla və ya boşluqlarla dolmasın, və (amma) kakimnibud başqa simvolla, onda ondan əvvəl (qarşısında) tək dırnağı qoyub istənilən simvolu

seçmək olar: ' #. ◆ spesifikatorun sol hissəsində % simvolu qoyulur, hansının ki, mövqeyindən və dəyişiklik başlanır.
Tələb edilən uzunluğu yalnızca ədədlərdə deyil, sətirlərdə tətbiq etmək olar. Bunun üçün müxtəlif əlavə edici simvolları və hətta düzəlişin ediləcəyi sol və ya sağ sərhədləri də seçmək olar. Mümkün variantlar nümunə 7.2-də göstərilmişdir.
Nümunə 7.2. Sətirlərin əlavəsi
<? php echo "<pre>"; // bütün boş sahələri əks etdirməyə icazə verən Teq
$h = 'Rasmus';
printf (" [%s] \n", $h); // printf (" [%12s] \n", $h) sətirinin Standart nəticəsi; // eni boyunca sağ künc üzrə boşluqların Düzləşdirməsi 12 printf (" [%-12s] \n", $h); // printf-in (" [%012s] \n", $h) sol kənarı (yeri) üzrə (görə) boşluqlarla Düzləşdirmə; // printf-in (" [%' # 12s] \n\n", $h) sıfırlarla Əlavəsi; // xüsusi olaraq seçilmişdən İstifadə // ' əlavəsinin simvolu # '
$d = 'Rasmus Lerdorf';
printf (" [%12.8s] \n", $d); // kəsilməylə sağ uc üzrə (görə) Düzləşdirmə // printf-in (" [%-12.12s] \n", $d) 8 simvoluna qədər; // kəsilməylə sol kənar (yer) üzrə (görə) Düzləşdirmə // printf-in (" [%-'@12.10s] \n", $d) 12 simvoluna qədər; // sol kənar (yer) üzrə (görə) Düzləşdirmə, əlavə // '@' simvoluyla, kəsilmə 10-a qədər simvol? >
Nəzərə alın ki, veb səhifədə lazımlı nişanın alınması üçün mən <pre> HTML-teqindən istifadə etmişəm. Bu teq, bütün boş sahələrə icazə verir və əks etdirilən hər bir sətirdən sonra \n yeni sətirinin simvolunu ekranlaşdırır. Bu nümunədə aşağıdakı mətn çıxarılır: [Rasmus] [Rasmus] [Rasmus] [00000 Rasmus] [# # # # # Rasmus] [Rasmus L] [Rasmus Lerdo] [Rasmus Ler@@]
Əgər işarə-yerlərin miqdarının göstərişi vaxtı (yanında) sətir uzunluğu artıq bu miqdara bərabərdirsə və ya onu ötür, bu göstəriş əhəmiyyət verilməyəcək, əgər simvolların yalnız verilmiş miqdarı, hansına ki, sətiri kəsmək lazımdır, znako-yerlərin daha az göstərilən miqdarı olmayacaq. Cədvəldə. 7.3 komponentlərə parçalanmış (yerləşdirilmiş) sətirin dəyişikliyinin spesifikatorları göstərilmişdir.

sprintf funksiyası

Elə hallar olur ki, proqramda nəticəni ekranlaşdırmaq deyil, yalnızca proqramın kodunda istifadə etmək zərurəti yaranır. Bunun üçün

`sprintf` funksiyası nəzərdə tutulmuşdur. Bu funksiya çıxış informasiyası brauzerə göndərilmir, lakin bu informasiyanı hər hansı bir dəyişənə mənimsətmək olar. Gəlin, əvvəlki nümunəyə sprintf funksiyalarını tətbiq edək. Nümunədə, **RGB** rəng uyğunluğu üzrə 65, 127, 245 qiymətləri onaltılıq say sistemində çevrilərək `$hexstring` dəyişəni mənimsədilir:

```
$hexstring = sprintf (" %X%X%X", 65, 127, 245);
```

Bu funksiyadan həmçinin çıxış informasiyasının saxlanılması üçün istifadə edilə bilər ki, istəsəz bu informasiyanı ekranlaşdıra da bilərsiniz:

```
$out = sprintf ("Nəticə: $%.2f", 123.42 / 12); echo $out;
```

Tarix funksiyaları və vaxt

Tarixin izlənilməsi və PHP-də vaxt üzrə hesablanma standartı UNIX vaxtına əsasən həyata keçirilir— 1 yanvar 1970-ci il. Cari vaxtın təyini üçün `time` funksiyasından istifadə etmək olar:

```
echo time();
```

Bir halda ki, vaxtın qiyməti saniyələr şəklində saxlanılır, həftə formasında olan vaxtın alınması üçün aşağıdakı ifadədən istifadə etmək tam düzgündür və burada qaytarılan qiymət ***7 gün × 24 saat × 60 dəqiqə × 60 saniyə*** artır:

```
echo time() + 7 * 24 * 60 * 60;
```

Əgər verilmiş tarix üçün vaxt qiymətini almaq lazımdırsa, mktime funksiyasından istifadə etmək olar. Bu funksiya, 2000-ci ilin birinci gününün, birinci saatının, birinci dəqiqəsinin birinci saniyəsi üçün vaxt 946684800 qiymətini çıxarır:

```
echo mktime (0, 0, 0, 1, 1, 2000);
```

Bu funksiyaya aşağıdakı parametrlər (soldan sağa) ötürülür:

- saat miqdarı (0 – 23);
- dəqiqə miqdarı (0 – 59);
- saniyə miqdarı (0 – 59);
- ayın sırası (1 – 12);
- ayın tarixi (1 – 31);
- il (1970 – 2038 və ya 1901 – 2038 [PHP 5.1.0-dan + rəqəm simvollarının 32 bitlik sistemlərindən istifadə etdikdə]).

Siz soruşa bilərsiniz: nəyə görə illər 1970 - 2038 aralığında məhdudlaşdırılmışdır? Bunun səbəb odur ki, UNIX-in ilk versiyasının istehsalçıları vaxt hesablamasının başlanğıcı kimi 1970-ci ili seçmişdilər, hansı ki, onların fikrincə, bu ildən daha əvvəl olan illər heç bir proqramçıya lazım olmayacaq. Xoşbəxtlikdən, PHP, 5.1.0 versiyasından başlayaraq, sistem, 32-dərəcəli tam ədəd işarələrindən istifadə edir, belə olan halda isə 1901-dən 2038-ci ilə qədər olan tarixlərin tətbiqi mümkün olur. Amma ikinci məhdudiyyət isə hələlik həllini tapmayıb. UNIX istehsalçıları əmin idilər ki, 70 il keçdikdən sonra heç kəs artıq onların sistemindən istifadə etməyəcək və buna görə də onlar, vaxt qiymətinin təyini üçün 32-bitlik yazılış (hansı ki, 19 yanvar 2038-ci ilə qədər məhdudiyyət şərti mövcuddur) üsulunu tətbiq ediblər. Bu məhdudiyyət Y2K38 adlandırılan kütləvi nasazlığa səbəb olacaq (oxşar problem 2000-ci ildə müşahidə edilmişdir, hansı ki, illərin qiymətləri iki rəqəm şəklində saxlanılırdı). Bu problemin həlli üçün PHP 5.2 versiyasına `DateTime` sinifi daxil edilmişdi, lakin bu sinif yalnız 64-bit arxitekturada işləyir.

Tarixin təsviri üçün bir çox formatlaşdırma tənzimlənmələrini dəstəkləyən `date` funksiyası istifadə olunur ki, bu halda tarix arzu olunan istənilən üsulla ekranlaşdırıla bilər. Bu funksiya aşağıdakı sintaksisə malikdir:

```
date ($format, $timestamp);
```

`$format` parametri sətir tipində olmalıdır. Bu parametrdə, cədvəldə təfərrüatı ilə təsvir edilmiş formatlaşdırma spesifikatorlarından istifadə edilir. `$timestamp` parametri UNIX standartında olan vaxt qiyməti olmalıdır. Spesifikatorların tam siyahısı http://php.net/manual/en/function.date.php ünvanında nəşr edilmişdir. Aşağıdakı komanda cari vaxtı və tarixi bu formatda ekranlaşdıracaq

```
"Thursday July 6th, 2017 - 1:38pm": echo date (" l F jS, Y - g:ia", time());
```

Sabitlər

Sabitlər, tarixlərlə bağlılar faydalı sabitlər Mövcuddurlar, hansılar ki, tarixlərlə bağlı komandalarla istifadə etmək olar ki, onlar müəyyən formatda tarixi qaytarsınlar. Məsələn, date (DATE_RSS) RSS-axında tətbiq edilən cari vaxtı və tarixi göstərir. Aşağıdakı sabitlər sıx istifadə edilən sabitlərdir.

• DATE_ATOM — Atom axınları üçün formatdır. "Y-m-d\TH:i:sP-i" PHP formatına bənzəyir və çıxarılan informasiya — "2018-08-16T12:00:00+0000".
• DATE_COOKIE — veb-serverlə və ya JavaScript ilə qurulan cookie üçün formatdır. "l-d-M-y H:i:s T" PHP-formatına bənzəyir və çıxarılan informasiya — "Thu, 16-Aug-2018 12:00:00 UTC".
• DATE_RSS — RSS axınları üçün formatdır. PHP-format bənzəyir, d M Y H:i:s T", və (amma) çıxarılan informasiya — "Thu, 16 Aug 2018 12:00:00 UTC".
• DATE_W3C — Ümumdünya hörümçək torunun Konsorsiumu üçün formatdır, World Wide Web Consortium. PHP-format "Y-m-d\TH:i:sP-i" bənzəyir, və (amma) çıxarılan informasiya — "2018-08-16T12:00:00+0000".

Tam siyahı http://php.net/manual/en/class.datetime.php ünvanında dərc edilmişdir.

Mümkün tarixi və vaxtı müxtəlif necə əks etdirməyin şahidi oldunuz. Kimi checkdate funksiyası, siz artıq gördünüz. *Bəs, verilən tarixin və vaxtın mümkünlüyünü necə yoxlamaq olar?* Bunun üçün `checdate` adlanan funksiyaya ayı, günü və ili daxil etmək lazımdır. Funksiya, mümkün tarix daxil etdikdə TRUE qiymətini, əks təqdirdə FALSE qiymətini qaytaracaq. Məsələn, əgər tarix istənilən ilin 30 fevralı daxil edilmişdirsə, əlbəttə ki, bu mümkünsüzdür. Nümunə 7.3-də kod göstərilmişdir bu funksiyadan necə istifadə etmək qaydası göstərilmişdir. Funksiya nümunədə göstərilən tarixin mümkünsüz olduğunu göstərir.

Nümunə 7.3. Tarixin mümkünlüyünün yoxlanılması

```
<?php
  $month = 9; // Sentyabr (hansı ki, bu 30 gündən ibarətdir)
  $day = 31; // 31-ci gün
  $year = 2018; // 2018
  if (checkdate ($month, $day, $year))
    echo "Mümkün tarix";
  else
    echo "Mümkünsüz tarix";
?>
```

Fayllarla iş

MySQL-ın bütün öz üstünlükləri vaxtı tək deyil (və ya ən yaxşı) veb-serverdə bütün məlumatların saxlanılmasının üsuluyla. Bəzən daha

sürətli və daha rahat bilavasitə diskdə saxlanılan fayllara müraciət etməyə zərurət yaranır. Bu təsvirlərin dəyişikliyi, məsələn istifadəçilər tərəfindən çıxardılmış avatarla və ya emal tələb edən qeyd jurnallarının faylları. Fayllar ilə əməliyyata başlamazdan əvvəl, faylların adlandırılmasına fikir verilməlidir. Əgər PHP-nin müxtəlif quraşdırmalarında istifadə oluna bilər kod yaradılırsa, onda o haqda bilmək (tanımaq), hərflərin registrinə sistem həssasdır ya yox, praktik olaraq mümkün deyil. Məsələn, Windows-da və Mac OS X-da fayl adları registrə qeyri-həssas, amma Linux-da və UNIX-də həssasdır. Buna görə də, belə hadisələrdən sığortalanmaq üçün sistemdən asılı olmayaraq registrə həssaslığa fikir verin və fayl adlarının kiçik hərf registrində olmasına tərəfdar olun. Faylın mövcudluğunun yoxlaması faylın mövcudluq faktını yoxlamaq üçün, file_exists funksiyasından istifadə etmək olar ki, bu funksiya da öz növbəsində TRUE və ya FALSE qaytarır və aşağıdakı qaydada istifadə olunur:

```
if (file_exists ("testfile.txt"))
      echo "Fayl mövcuddur";
```

Faylların yaradılması

İndi testfile.txt faylı mövcud deyil, buna görə onu yaradacağıq və yaradacağımız fayla bir neçə sətir yazacağıq. Nümunə 7.4-də göstərilmiş kodu daxil edin və onu testfile.php adının saxlayın.

Nümunə 7.4. Sadə mətn faylının yaradılması

```
<? php
// testfile.php
$fh = fopen ("testfile.txt", 'w') or die (" faylı Yaratmaq müvəffəq olmadı");
  $text = <<<_END Sətir 1 Sətir 2 Sətir 3
  _ END;
fwrite ($fh, $text) or die (" faylın yazısının Nasazlığı");
fclose ($fh);
echo "'testfile.txt' Faylı müvəffəqiyyətlə yazılmış ";
? >
```

Əgər bu kod brauzer vasitəsilə icra olunacaqsa, onda kodun müvəffəqiyyətli icrası zamanı mesaj ekranlaşdırılacaq: "'testfile.txt' faylı müvəffəqiyyətlə yazılmış". Əgər xəta barədə məlumat ekranlaşdırılarsa, deməli, diskdə kifayət qədər yer yoxdur və ya çox ehtimal ki, sistem tərəfindən fayl yaradılmasına və ya fayla yazılışına icazə verilmir. Bu halda siz əməliyyat sisteminiz tələblərinə uyğun olaraq təyinat qovluğunun atributlarını dəyişdirməlisiniz. Əgər əməliyyat xətasız başa çatsa, onda testfile.txt faylı testfile.php faylının

yerləşdiyi qovluğa düşəcək. Əgər mətn və ya proqram redaktorunda faylı açsaq, onda fayl aşağıdakı tərkibdə olacaq: Sətir 1 Sətir 2 Sətir 3
bu sadə nümunədə bütün fayllarla iş ardıcıllığı göstərilmişdir.
1. Hər şey fopen funksiyanın köməyi ilə faylın açılışından başlayır.
2. Bundan sonra başqa funksiyalardan etmək olar. İndiki halda fayla yazı (fwrite) əməliyyatı aparılır, amma bundan başqa artıq mövcud olan fayldan məlumatları oxumaq (fread və ya fgets) və faylla başqa əməliyyatları da həyata keçirmək olar.
3. Proses faylın bağlanmasıyla yekunlaşır.
Hərçənd ki, proses bitdikdən proqram özü avtomatik olaraq faylı bağlayır, amma yenə də prosesin sonunda fayl bağlanacağına əmin olmaq üçün faylı bağlamağınız məsləhətdir. Fayl resursu hər bir açıq fayla tələb olunur ki, PHP-proqram ona müraciət edə və onu idarə edə bilsin. `$fh` (hansını ki, mən faylın təsvir edəcisi kimi seçdim) dəyişəninin əvvəlki nümunəsində `fopen` funksiyasıyla qaytarılan qiymət mənimsənilir. Bundan sonra hər bir fayl emalı funksiyasına emal edilən faylı müəyyən etmək üçün `$fh` dəyişəni parametr olaraq daxil edilir. `$fh` dəyişənin tərkibində qeyri-adi heç nə yoxdur və bu yalnız fayl haqqında daxili informasiyaya istinad üçün PHP tərəfindən istifadə edilən nömrədir. Bu dəyişən yalnız başqa funksiyalara ötürülmə üçün istifadə olunur. Nasazlıq halında `fopen` funksiyası `FALSE` qiymətini qaytarır.
Əvvəlki nümunədə nasazlığın tənzimlənməsi və reaksiyasına aid sadə üsul göstərilmişdir: nasazlıq zamanı die funksiyası çağırılır ki, bu funksiyada öz növbəsində proqramı yekunlaşdırır və istifadəçiyə xəta barədə məlumat verir. `fopen` funksiyası çağırışında istifadə edilən ikinci parametrə diqqət yetirin. Bu funksiya yazı üçün faylı açmaq əmri verən `w` simvoludur. Əgər belə adda fayl yoxdursa, onda bu adda yaradılacaq. Bu funksiyanı tətbiq edərkən ehtiyatlı olmaq lazımdır: əgər fayl artıq mövcuddursa, onda `w` iş rejimi parametri fopen funksiyasını faylın əvvəlki tərkibini tam olaraq silməyə məcbur edəcək (hətta əgər yeni faylın məzmun boş olsa belə!).
Cədvəl 7.5-də bu funksiyanın çağırışı istifadə edilən müxtəlif iş rejiminin parametrləri göstərilmişdir.
Таблица 7.5. Режимы работы, поддерживаемые функцией fopen
Режим Действие Описание 'r' Чтение с начала файла Открытие файла только для чтения; установка указателя файла на его начало. Возвращение FALSE, если файла не существует 'r+' Чтение с начала файла с возможностью записи Открытие файла

для чтения и записи; установка указателя файла на его начало. Возвращение FALSE, если файла не существует 'w' Запись с начала файла с усечением его размера Открытие файла только для записи; установка указателя файла на его начало и сокращение размера файла до нуля. Если файла не существует, попытка его создания 'w+' Запись с начала файла с усечением его размера и возможностью чтения Открытие файла для чтения и записи; установка указателя файла на его начало и сокращение его размера до нуля. Если файла не существует, попытка его создания 'a' Добавление к концу файла Открытие файла только для записи; установка указателя файла на его конец. Если файла не существует, попытка его создания 'a+' Добавление к концу файла с возможностью чтения Открытие файла для чтения и записи; установка указателя файла на его конец. Если файла не существует, попытка его создания

Чтение из файлов Проще всего прочитать текстовый файл, извлекая из него всю строку целиком, для чего, как в примере 7.5, используется функция fgets (последняя буква s в названии функции означает string — «строка»). Пример 7.5. Чтение файла с помощью функции fgets <?php $fh = fopen("testfile.txt", 'r') or die("Файл не существует, или вы не обладаете правами на его открытие");

$line = fgets($fh); fclose($fh); echo $line; ?> Если используется файл, созданный кодом из примера 7.4, будет получена первая строка: Строка 1 Можно также извлечь из файла сразу несколько строк или фрагменты строк, воспользовавшись функцией fread, как показано в примере 7.6. Пример 7.6. Чтение файла с помощью функции fread <?php $fh = fopen("testfile.txt", 'r') or

die("Файл не существует, или вы не обладаете правами на его открытие");

$text = fread($fh, 3); fclose($fh); echo $text; ?> При вызове функции fread было запрошено чтение трех символов, поэтому программа отобразит следующий текст: Стр Функция fread обычно применяется для чтения двоичных данных. Но если она используется для чтения текстовых данных объемом более одной строки, следует брать в расчет символы новой строки.

Faylların kopyalanması

`copy` funksiyasından istifadə edib bizim `testfile.txt` faylının klonunu yaratmağa cəhd edək. Nümunə 7.7-dəki mətni daxil edin və copyfile.php kimi yadda saxlayın. Sonra isə, brauzer vasitəsilə proqramı açın.

Nümunə 7.7. *Faylın kopyalanması*

```
<?php
  //copyfile.php
  copy ('testfile.txt', 'testfile2.txt') or die ("Kopyalama mümkün
    deyil");
  echo "Fayl 'testfile2.txt' adlı fayla uğurla kopiyalanmışdır";
?>
```

Əgər proqramın icrasından sonra qovluğunuzun tərkibini yoxlasaq, onda testfile2.txt adlı yeni faylın yaradıldığının şahidi olacağıq. Yeri gəlmişkən, əgər siz uğursuz cəhdindən sonra proqramın öz işini bitirməsini istəmirsinizsə, başqa bir sintaksis variantından istifadə etmək olar. Məsələn, nümunə 7.8-də nəzər yetirin.

Nümunə 7.8. *Faylın kopyalanması üçün alternativ sintaksis*

```
<?php
  //copyfile2.php
  if (! copy ('testfile.txt', 'testfile2.txt'))
     echo "Kopyalama mümkün deyil";
  else
     echo "Fayl 'testfile2.txt' adlı fayla uğurla kopiyalanmışdır";
?>
```

Faylın yerdəyişməsi

Faylın yerdəyişməsi üçün nümunə 7.9-da göstərildiyi kimi, faylın adını dəyişdirmək lazımdır.

Nümunə 7.9. Faylın yerdəyişməsi

```
<?php
  //movefile.php
  if (!rename ('testfile2.txt', 'testfile2.new'))
     echo "Faylın adını dəyişdirmək mümkün deyil";
  else
     echo "Faylın adı 'testfile2.new' formasında dəyişdirilmişdir";
?>
```

Ad dəyişdirmə funksiyasını qovluqlara da tətbiq etmək olar. Faylların mövcud olmaması xətasından sığortalanmaq üçün, əvvəlcə faylların mövcudluq faktının yoxlanılması üçün file_exists funksiyasından istifadə etmək olar.

Fayl sistemindən faylın silinməsi

Üçün faylın silinməsi (uzaqlaşdırması) kifayətdir, göstərildiyi kimi nümunə 7.10-da, bunu etməyə icazə verən unlink funksiyasından istifadə etmək. Nümunə 7.10. Faylın silinməsi <? php // deletefile.php if (! unlink ('testfile2.new')) echo "Silinmə (Uzaqlaşdırma) "; else echo "'testfile2.new' Faylı müvəffəqiyyətlə silinmiş (uzaqlaşdırılmış)" mümkün deyil;? >

Sərt diskdə olan fayllara bilavasitə giriş zamanı, əməliyyatların sizin fayl sisteminizi təhlükə altına qoymayacağına əmin olun. Məsələn, informasiyanın istifadəçisi tərəfindən mütləq onda arxayın lazım olmağa daxil edilmiş əsasında faylın silinməsi (uzaqlaşdırması) vaxtı ki, bu fayl ziyansız sistemin təhlükəsizlikləri silinmiş (uzaqlaşdırılmış) ola bilər və istifadəçiyə onu uzaqlaşdırmağa icazə verilmişdir (həll edilmişdir).

İndiki halda, necə ki, yerdəyişmənin əməliyyatı vaxtı, əgər belə adla fayl mövcud deyilsə, xəbərdarlıq çıxarılacaq, hansının ki, yaranmaları (meydana çıxmaları) qaçmaq olar, əgər unlink funksiya çağırışından əvvəl (qarşısında) onun mövcudluğunun yoxlaması üçün file_exists funksiyasından istifadə etməksə.

Faylların yenilənməsi

ehtiyac saxlanmış fayla əlavə məlumatları əlavə etməyə Kifayət qədər tez-tez yaranır, nə üçün bir çox üsul mövcuddur. Məlumatların əlavə edilməsinin rejimlərinin birindən istifadə etmək olar (cədvələ baxmaq. 7.5) və ya yazını dəstəkləyən rejimi cəlb etmək və oxuma və yazı üçün faylı açmağa və o yerə faylın göstəricisinin (siyahısının) yerini dəyişməyə (köçürtməyə) sadədir, hansından ki, fayldan fayla yazını və ya oxumanı aparmaq lazımdır. Faylın göstəricisi — oxuma və yazılış zamanı fayla girişi təmin faylın daxilindəkı mövqedir. Göstəricini, fayl haqqında məlumatları ehtiva edən (hansı ki, nümunə

7.4-də $fh dəyişənində saxlanılırdı) faylın təsviri qarışdırmayın. Əgər nümunə 7.11-də göstərilmiş kodu update.php kimi faylı yadda saxlayıb icra etsək, nə olduğunu anlayacaqsınız. Nümunə 7.11. Faylın yenilənməsi <? php // update.php $fh = fopen (" testfile.txt", 'r+') or die (" faylın açılışının Nasazlığı"); $text = fgets ($fh);
fseek ($fh, 0, SEEK_END);
fwrite ($fh, "$text") or die (" fayla yazının Nasazlığı"); fclose ($fh);
echo "'testfile.txt' Faylı müvəffəqiyyətlə yenilənmiş";? > Bu proqram '+r' iş rejimini göstərildiyinə görə testfile.txt faylını oxuma və yazı üçün açır ki, burada göstərici faylın başlanğıcı müəyyən edilir. Sonra fgets funksiyasından istifadə olunur ki, fayldan bir sətir (simvol ehtiva edən ilk sətir) oxunur. Bundan sonra fseek funksiyası çağırılır. Bu funksiya faylın göstəricisini yerini dəyişmək ən sona üçün keçirir ki, sonra da bura faylın başlanğıcından çıxardılmış sətir ($text dəyişəninə mənimsədilmiş) əlavə edilir. Sonda isə, fayl bağlanır. Nəticədə alınmış fayl aşağıdakı məzmuna malik olur: Sətir 1 Sətir 2 Sətir 3 Sətir 1 Birinci sətir müvəffəqiyyətlə kopiyalanmış və faylın sonuna əlavə edilmişdir. Fseek funksiyasının bu nümunəsində, $fh faylının təsvirindən başqa, daha iki parametr əlavə edilmişdir — 0 və SEEK_END. SEEK_END parametri funksiyaya fayl göstəricisinin yerini dəyişmək ən sona keçirtməyi əmr verir. 0 parametri isə, göstərir, geri qayıtmaq mövqeyini lazımdır. Nümunə 7.11-də bu parametrin qiymət 0 olaraq təyin edilir, çünki göstərici faylın sonunda qalmalıdır. Fseek funksiyasında göstərici nizamlanması üçün daha iki rejimdən istifadə olar: SEEK_SET və SEEK_CUR. SEEK_SET rejimi funksiyanı fayl göstəricisini əvvəlki parametrlə verilmiş konkret mövqeyə nizamlanmasını əmr verir. Buna görə də aşağıdakı nümunədə faylın göstəricisinin mövqeyi 18-ə dəyişdirilir: fseek ($fh, 18, SEEK_SET); SEEK_CUR Rejimi faylın göstəricisinin cari mövqedən yerinin dəyişdirilməsini nizamlayır. Əgər indi fayl göstəricisi mövqeyi 18-dirsə, onda aşağıdakı növdə funksiya çağırışı onun mövqeyi 23-ə dəyişdirəcək: fseek ($fh, 5, SEEK_CUR); xüsusi ehtiyacsız bunu Etmək tövsiyə edilmir, amma beləliklə hətta mətn faylları (sətirlərin bərkidilmiş uzunluğuyla) sadə nestrukturirovannıx məlumat bazaları kimi istifadə etmək olar. Bu halda sizin proqramınız çıxartma üçün belə fayl üzrə (görə) hər iki tərəfə yerdəyişmə üçün fseek funksiyasından istifadə edə bilər, mövcud olanların yenilənmələri və yeni yazıların əlavə edilmələri. Yazılar

həmçinin onların sıfır simvollarla yenidən yazması yolu ilə uzaqlaşdırıla (silinə) bilər və s.

Fayllara kütləvi müracətlər zamanı blokirovkalar

Veb-proqramlara kütləvi daxilolmalar zamanı faylların mühasirə edilməsi eyni zamanda çox istifadəçilər tərəfindən kifayət qədər tez-tez çağırılırlar. Birdən çox istifadəçinin eyni zamanı fayl yazılışı əməliyyatını həyata keçirməsi fayla zərər vura bilər. Və bir istifadəçi fayla yazı əməliyyatı aparırsa, başqası bu fayldan məlumatlar oxuyursa, faylla heç nə olmayacaq, amma faylı oxuyan olduqca qəribə nəticələri ala bilər.Bir neçə istifadənin eyni zamanda fayla müraciət etməsini nizamalamaq üçün, fayl blokirovkası funksiyası olan flock-dan istifadə etmək lazımdır. Bu funksiya yalnız proqramınız blokirovkanı götürdükdən sonra fayla edilən sorğulara cavab verir və bir növ fayla girişi növbəlilik prinsipi ilə tənzimləyir. Sizin proqramlarınız eyni zamanda bir istifadəçiyə də əlçatan olan fayl(lar)a müraciət etdikdə, niyyətlə ona yazını edə bilər, koda həmçinin faylın blokirovkasına tapşırığı əlavə etmək lazımdır, necə nümunə 7.12-də, hansı ki, nümunə 7.11 yenilənmiş versiyasıdır. Nümunə 7.12. Blokirovkadan istifadəylə faylın yenilənməsi <? php $fh = fopen (" testfile.txt", 'r+') or die (" faylın açılışının Nasazlığı"); $text = fgets ($fh);

if (flock ($fh, LOCK_EX)){ fseek ($fh, 0, SEEK_END); fwrite ($fh, "$text") or die (" fayla yazının Nasazlığı"); flock ($fh, LOCK_UN);}

fclose ($fh); echo "'testfile.txt' Faylı müvəffəqiyyətlə yenilənmiş";? > sizin saytınızın ziyarətçiləri üçün faylın blokirovkası ən kiçik reaksiya vaxtı qurulmalıdır: blokirovkanı fayla dəyişikliklərin daxil edilməsindən bilavasitə qabaq qoymaq və əməliyyat bitdikdən blokirovkadan çıxartmaq lazımdır. Faylın blokirovkasının uzun müddət olması proqramın işinin əsassız ləngiməsinə gətirib çıxaracaq. Buna görə də nümunə 7.12-də flock funksiyası bilavasitə fwrite funksiyasının çağırışından əvvəl və sonra çağırılır. LOCK_EX parametrinin köməyi ilə flock-un birinci çağırışı zamanı faylın ilk blokirovkası qurulur ki, istinad $fh dəyişənindən götürülür: flock ($fh, LOCK_EX); bu andan etibarən fayl üzərində yalnız yazı deyil heç bir proses həyata keçirilə bilmir. Ancaq funksiyaya LOCK_UN parametrinin ötürülməsinin sonra fayl blokirovkadan çıxır: flock ($fh, LOCK_UN); blokirovkadan çıxan kimi, fayla giriş bütün proseslərə

yenidən əlçatan olur. Hər oxuma və ya məlumatların yazılışı zamanı fayla yenidən müraciət etmək lazımdır: fayla son müraciət vaxtından başqa proses dəyişikliyin bu faylına daxil edə bilirdi. Yeri gəlmişkən, siz qeyd edə bilərsiniz ki, ilk blokirovka tələbi niyə if təlimatı ilə verilmişdir? Məsələ ondadır ki, flock bütün sistemlərdə dəstəklənmir və buna görə də blokirovka quraşdırılmasının mümkünlüyünü yoxlamaq lazımdır. Həmçinin yadda saxlamaq lazımdır ki, flock funksiyası tövsiyə edilən blokirovkaya aiddir. Bu onu bildirir ki, yalnız funksiya çağıran proseslər bloklanır. Əgər kodunuz hər hansı hissə fayllara birbaşa təsir edirsə və dəyişdirirsə, zaman keçdikcə fayllarınızda xaos vəziyyəti yaranacaq. Lakin flock funksiyadan istifadə etsəz, bu proseslərə tam nəzarət edəcəksiniz. Yeri gəlmişkən, əgər hansısa bir kod fraqmentində fayl bloklasaz, sonda isə blokirovkanı qaldırmağı yaddan çıxarsaz aşkar edilməsi çətin olan xətalar ilə üzləşəcəksiniz. Ona görə diqqətli olun.

`flock` funksiyası şəbəkə fayl sistemində (NFS) və şəbəkələrin tətbiqinə əsaslanan bir çox başqa fayl sistemlərində işləmır. ISAPI tipində çoxaxınlı serverlərindən istifadə edərkən `flock`-a güvənməyin, çünki `flock`, faylları fiziki serverin paralel axınlarda realizasiya olunan PHP-ssenarilərin kodundan qorumayacaq. Bundan başqa, `flock` köhnəlmiş fayl sistemi olan FAT-dan istifadə edən istənilən sistemlərdə məsələn Windows-un köhnəlmiş versiyalarında dəstəklənmir.

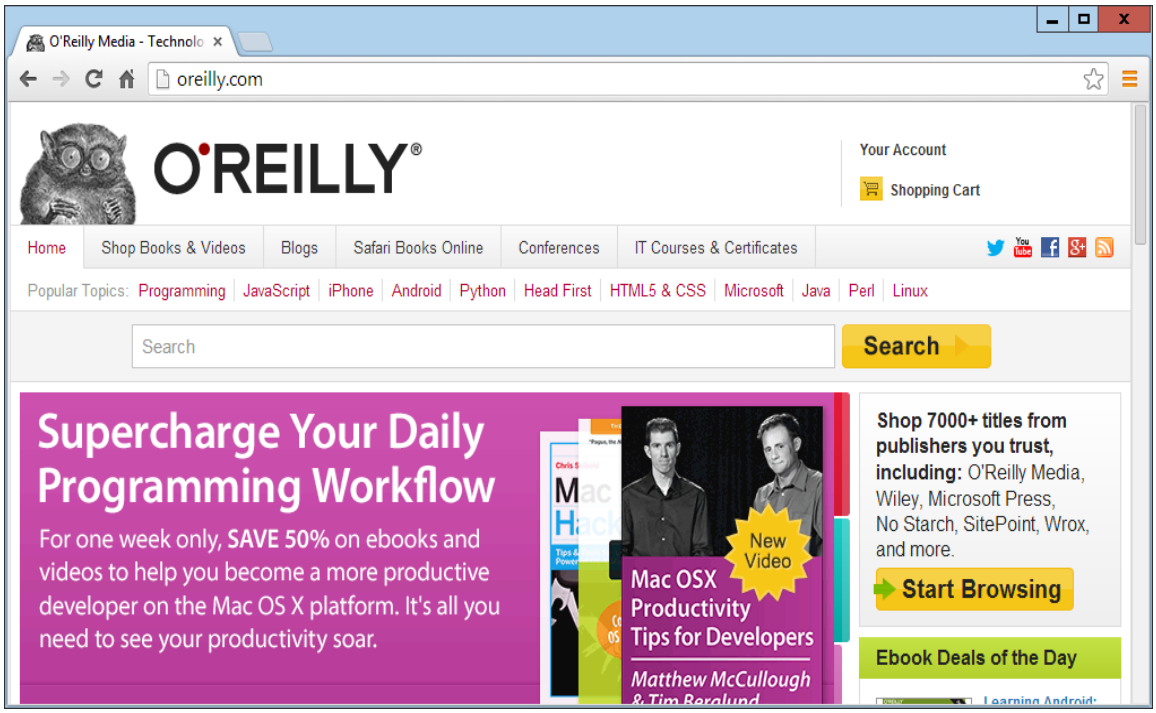
Faylın tərkibinin tam oxunması

Bütün faylın oxunması faylların təsvir edənlərindən istifadəsiz tamamilə bütün faylın oxunması Üçün file_get_contents-in çox rahat funksiyasından istifadə etmək tamamilə olar. Bu funksiyanın tətbiqi çox sadədir və nümunə 7.13 göstərilmişdir. Nümunə 7.13. File_get_contents funksiyasından istifadə <? php echo "<pre>"; // Teq, echo file_get_contents (" testfile.txt") sətirlərinin tərcümələrini

(köçürmələrini) əks etdirməyə icazə verən; echo "</pre>"; // pre teqinin hərəkətinin (təsirinin) Dayandırılması? > Amma bu funksiyanın əsas bacarığı İnternet vasitəsilə serverdən faylı çıxarta bilməsidir. Nümunə 7.14-də sonrakı onun təsviriylə O'Reilly saytının əsas səhifəsindən HTML kodunun sorğusu göstərilmişdir, necə özünə adi keçid vaxtı veb səhifəni. Alınmış nəticə şəkildə göstərilən səhifənin surətinə bənzəyəcək. 7.1. Nümunə 7.14. O'Reilly saytının əsas səhifəsinin tutulması <? php echo file_get_contents (" http://oreilly.com");? >

Veb-serverə faylların yükləməsi

veb-serverə faylların Yükləməsi zamanı bir çox insanlar çətinliklərə səbəb olur (çağırır), amma bu prosesi daha sadə etmək, nə qədər o əslində var, mümkün təqdim edilmir (təsəvvür edilmir). Faylın formadan yüklənməsi üçün yalnız kodlaşdırmanın xüsusi tipini yəni, multipart/form-data tipini seçmək lazımdır. Qalan işi isə sizin brauzeriniz edəcək. Bu prosesi praktiki görmək üçün, nümunə 7.15-də təqdim edilmiş proqramı upload.php kimi yadda saxlayın. Bu faylın icrası brauzer vasitəsilə istənilən faylı serverə yükləmək olar.



Şəkil. 7.1. `file_get_contents` *funksiyasının köməyi ilə tutulmuş O'Reilly saytının əsas səhifəsi*

Nümunə 7.15. upload.php faylında saxlanılan təsvirlərin yükləməsi üçün proqram <? php // upload.php echo <<<_END server</title> </head><body>-da faylların yükləməsi üçün <html><head> <title>PHP-forma <form method='post' action='upload.php' enctype='multipart/form-data'>-ı faylı Seçin: <input type='file' name='filename' size='10'> <input type='submit' value='Zaqruzit'> </form> _END;

if ($_FILES){ $name = $_FILES ['filename'] ['name']; move_uploaded_file ($_FILES ['filename'] ['tmp_name'], $name); "'$name'
<img src='$name'>-in Yüklənən təsviri" echo-u;}

echo "</body></html>";? > Proqramı bloklar üzrə təhlil edək. Echo-un multixətt təlimatının birinci sətirində HTML-sənədin başlanğıcı, başlığı əks olunur və sonra sənədin gövdəsi başlanır.

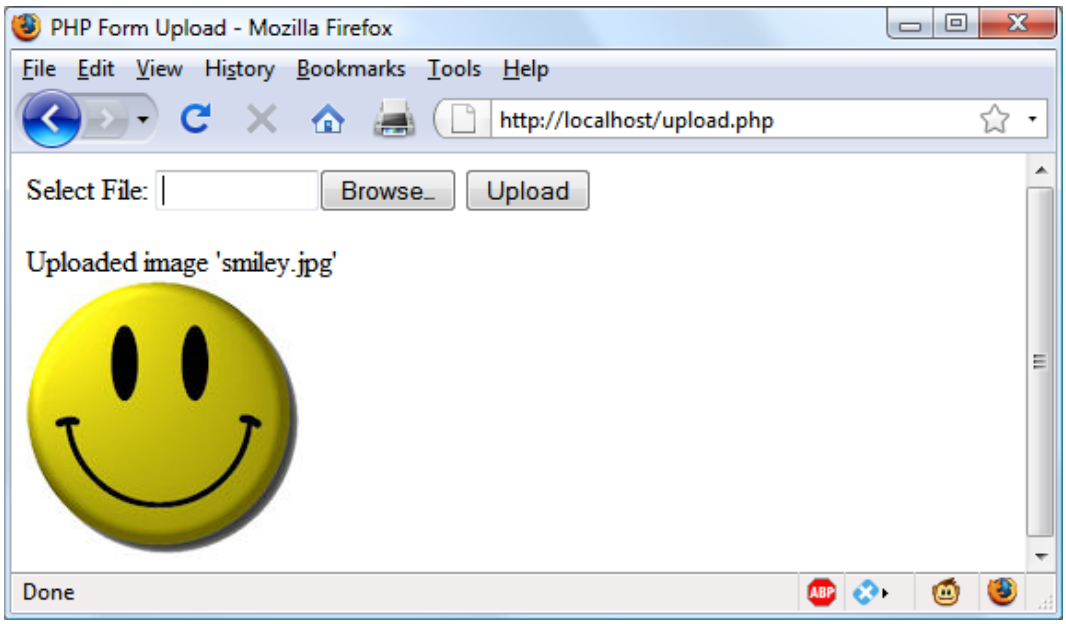
Sonra tərkibinin ötürülməsi üçün POST metodu seçilmiş, bütün göndərilənlərin upload.php (yəni proqramın özünə) proqramına təyin edilmiş və göndərilən məlumatlar multipart/form-data tərkibinin tipindən istifadə etməklə kodlaşdırılmış forma yerləşdirilir. Aşağıdakı sətirlərdə isə formanın doldurulması üçün elementlər yerləşdirilir və istifadəçidən iki dəfə seçim etməsi istənilir. Əvvəlcə istifadəçidən faylı göstərmək tələb olunur. Daxil parametrlərində file informasiyanın tipini (input type), filename faylın adını, size isə daxiletmə sahəsinin ölçüsünü göstərir. (Bu yazı susmaya görə istifadə edilən mətni əvəz edir sonra istifadəçidən göndərilməyə komandanı bu formaları daxil etmək tələb olunur, nə üçün yazıyla düymə Yükləməyə xidmət edir — Submit Query, "sorğunu Göndərmək" nəyi bildirir). Bundan sonra forma bağlanır. Bu kiçik proqramda veb-proqramlaşdırmanın olduqca geniş yayılmış texnologiyası göstərilmişdir ki, eyni proqram iki dəfə çağırılır: birinci dəfə səhifənin ilk ziyarət zamanı, ikinci dəfə isə, istifadəçi formanın göndərilməsi düyməsini basdığı anda. Yüklənən məlumatların qəbulu üçün nəzərdə tutulmuş PHP-kod son dərəcə sadədir, çünki, serverə yüklənən yüklənən fayllar $_FILES assosiativ sistem massivində yerləşir. Buna görə də faylın istifadəçi tərəfindən göndərilməsi faktının nizamlanılması üçün, hər hansı məzmunun $_FILES massivində mövcudluğunu yoxlamaq kifayətdir. Bu yoxlama if ($_FILES) təlimatının köməyi ilə həyata keçirilir. İstifadəçi tərəfindən səhifənin birinci ziyarəti zamanı, faylın yükləməsinə qədər $_FILES massivi boş olur, buna görə də proqram kodun bu blokunu buraxır. İstifadəçi fayl yükləyərsə, proqram yenidən icra olunur və

$_FILES massivində elementin mövcudluğu aşkar edilir. Proqram fayl yüklənməsini aşkar etdikdə, yüklənildiyi mənbədən faylın adını $name dəyişəninə ötürür. İndi yalnız faylın yerini PHP-nin yüklənmiş faylları saxladığı müvəqqəti yerdən daimi saxlama yerinə dəyişmək lazımdır. Bu proses move_uploaded_file funksiyasının köməyi ilə icra edilir ki, bu funksiyaya cari kataloqda (qovluqda) saxlanan yülənən faylın ilk adı ötürülür. Və nəhayət, serverə yüklənmiş təsvirin IMG teqində qeyd etməklə, təsvir əks olunur. Mümkün nəticə şəkildə göstərilmişdir. 7.2.

*Əgər proqramın işə salınması zamanı move_uploaded_file funksiyasının çağırışına cavab olaraq giriş hüquqlarının yoxluğu — **Permission denied** xəbərdarlığı meydana çıxsa, deməli proqramın icra olunduğu giriş hüququnuz yoxdur.*

$_FILES massivindən istifadə

serverə faylın yükləməsi Vaxtı $_FILES massivindən istifadə cədvəldə göstərilmiş beş element qalır (saxlanır). 7.6 (harada yüklənən fayl istifadə olunur, hansının ki, adı serverə göndərilən formayla verilir).



Şəkil. 6.2. *Məlumatların formasının köməyi ilə təsvirin yükləməsi*

Cədvəl 7.6. $_FILES massivinin tərkibi

Məzmun tiplərini adətən MIME-tip (Multipurpose Internet Mail Extension — İnternetdə çoxməqsədli poçt genişlənmələri) adlandırırdılar. Amma sonralar informasiyanın bütün növləri İnternet yayıla bildiyinə görə, bu məzmun tiplərini İnternetdə istifadə edilən informasiya tipi (Internet media types) kimi adlandırırlar. Cədvəldə. 7.7 $_FILES ['file'] ['type'] massivinin elementində meydana çıxan ən tez-tez istifadə edilən tiplərdən bəziləri göstərilmişdir.

Cədvəl 7.7. Bəziləri İnternetdə istifadə edilən informasiyanın ən yayılmış tipləri

Fayl tiplərinin nizamlanması

Yəqin bilirsiniz ki, əksər kiber-hücumlar "qurban" serverə yüklənmiş fayllar həyata keçirilir.

Daxil edilmiş məlumatların zərərliliyini, həmçinin fayl artıq yükləndiyini daim yoxlamaq lazımdır. Əgər fayl artıq yüklənmişdirsə, onda faylın düzgün məlumat tipi göndərilmiş olduğunu yoxlamaq lazımdır. Bütün bunları nəzərə alaraq upload.php proqramı nümunə 7.16-da göstərilmiş formada upload2.php proqramına çevrilib. Nümunə 7.16. Upload.php-nin daha təhlükəsiz versiyası <? php // upload2.php echo <<<_END server</title></head><body>-da faylların yükləməsi üçün <html> <head><title>PHP-forma <form method='post' action='upload2.php' enctype='multipart/form-data'>-ı JPG, GIF, PNG və ya TIF genişlənilməsiylə faylı Seçin: <input type='file' name='filename' size='10'> <input type='submit' value='Zaqruzit'></form> _END;

if ($_FILES){ $name = $_FILES ['filename'] ['name'];

switch ($_FILES ['filename'] ['type']){ case 'image/jpeg': $ext = 'jpg'; break; case 'image/gif': $ext = 'gif'; break; case 'image/png': $ext = 'png'; break; case 'image/tiff': $ext = 'tif'; break; default: $ext = ''; break;} if ($ext){ $n = "image.$ext"; move_uploaded_file ($_FILES ['filename'] ['tmp_name'], $n); echo "'$n':
 adının altında '$name' təsviri Yüklənmiş"; echo "<img src='$n'>";} else echo "'$name' — təsvirin qəbuledilməz faylı";} else echo "təsvirin Yükləmələri olmadı";

echo "</body></html>";? > burada HTML-kod ehtiva etməyən blok genişləndirilmişdir. Proqramın əvvəlki versiyasında olduğu kimi, if təlimatı vasitəsilə məlumatların göndərilməsinin faktının

yoxlanılmasını yerinə yetirilir, amma bu versiyada kodun sonuna yaxın if təlimatın daxilində yenidən if/else kontrolu yerinə yetirilir. Həmçinin əsas if təlimatına da else kontrolu əlavə edilmişdirki, bu kontrol uğursuzluq zamanı xəbərdarlıq mesajını ekranlanlaşdırır.
if təlimatının daxilində mənbə kompüterdən alınmış faylın adı $name dəyişəninə mənimsədilir, amma bu dəfə istifadəçinin bizə münasib məlumatları göndərdiyinə güvənmirik. Buna görə də yüklənən kontekstin tipinin uyğunluğunun yoxlanılması üçün, öncə faylın tipi switch təlimatından keçir. swich təlimatına dəstəklənən dörd təsvir formatı nəzərdə tutulmuşdur. Yüklənən fayl təlimatdakı uyğunluğa cavab verirsə fayl tipi $ext dəyişəninə mənimsədilir. Əgər uyğunluq müəyyən edilməsə, deməli, yüklənmiş fayl münasib tipə aid deyil və $ext dəyişəninə boş sətir mənimdiləcək "". Növbəti blokda isə $ext dəyişəni sətir ehtiva etdiyi yoxlanılır, müsbət cavab gələrsə $n dəyişəninə "image.$ext" formatında mətn mənimsədilir və bu mətn faylın yeni adı olur. Bu onu bildirir ki, proqram yaradılan faylın adına tamamilə nəzarət edir və bu adla yalnız bu fayllar ola bilər: image.jpg, image.gif, image.png və ya image.tif. İndi proqram tam təhlükəsiz icra olunur. Qalan PHP-kodu əvvəlki versiyanın koduna oxşardır. O onun yeni yerinə yüklənmiş müvəqqəti təsviri qarışdırır, sonra onu ekrana çıxarır, və (amma) onunla birlikdə təsvirin köhnə və yeni adlarını əks etdirir.

Yükləmə prosesində PHP-də yaradılmış müvəqqəti faylın silinməsi barədə narahat olmaq lazım deyil, çünki, əgər faylın yeri və adı dəyişilməmişsə, proqram bağlandığı anda fayl avtomatik silinəcək.

if təlimatının şərtinə görə else təlimatına keçid olan zaman, məsələn əgər dəstəklənməyən təsvir tipi yüklənmişdirsə, proqram xəta barədə məlumatı çıxarır. Mən eyni yanaşmanı yüklənən fayllar üçün də etməyi məsləhət görürəm. Bunun üçün qabaqcadan seçilmiş adlardan və yerlərdən istifadə etməyi siz yükləmənin şəxsi proqramını yaradacaqsınız. Onda hər hansı başqa yol adlarının istifadə edilən dəyişənlərinə əlavə edilmənin istənilən cəhdləri çıxarılacaq (istisna

ediləcək) və başqaları bunları, bacarıqlıları ziyanı gətirmək (vurmaq). Əgər nəzərdə bir neçə istifadəçi eyni adlı faylı yükləməsi ehtimalı varsa, onda belə fayllar istifadəçiləri adlarında təşkil edən prefikslərlə təchiz etmək və ya bu hər bir istifadəçi üçün yaradılmış xüsusi qovluqlarda saxlamaq olar. Amma əgər verilmiş faylın adından istifadə etmək lazımdırsa, bu faylı zərərsizləşdirmək lazım olacaq, bunun üçün müntəzəm ifadələr vasitəsilə yalnız hərf-rəqəm simvollarının və nöqtənin tətbiqinə icazə verib, əvəzetmə əməliyyatını $name dəyişəninə mənimsədirik: $name = ereg_replace (" [^A-Za - z0-9.]", "", $name); Bu komanda görə $name dəyişənindəki sətir yalnız A – Z, a – z, 0 – 9 və nöqtə olmalıdır, bu simvollar qalan bütün simvollar uzaqlaşdırılır. Bütün sistemlərdə çalışın ki, sistemin reqistrə həssaslığından asılı olmayaraq proqramınızın işini kiçik hərf reqistri ilə qurun: $name = strtolower (ereg_replace (" [^A-Za - z0-9.]", "", $name));

Bəzən `image/pjpeg` *məzmun tipi ilə qarşılamaq olar. Bu format proqressiv JPEG-formatı adlanır. Bu tipi kodunuza o biri formatlar kimi əlavə edib, təhlükəsiz yükləyə bilərsiniz:*
case 'image/pjpeg':
case 'image/jpeg': $ext = 'jpg'; break;

Sistem çağırışları

Bəzən PHP-nin çalışdığı əməliyyat sistemində konkret işin həyata keçirilməsi üçün PHP-funksiyası tapılmır. Bu halda məsələnin icrası üçün exec sistem çağırışını tətbiq etmək olar. Məsələn, cari kataloqun tərkibinə baxış üçün nümunə 7.17-də göstərilmiş proqramdan istifadə etmək olar. Windows sistemində kodda heç bir dəyişikliklərə ehtiyac yoxdur və Windows dir komandasını icra edəcək. Linux-da, UNIX-də və ya Mac OS X-da ls komandasını icra etmək üçün birinci sətiri və ümumilikdə şərhləri silmək lazımdır. İstəkdən asılı olaraq bu proqramın mətnini exec.php kimi saxlaya və onu brauzerindən icra edə bilərsiniz. Nümunə 7.17. Sistem komandasının icrası (ifası) <?

php // exec.php $cmd = "dir"; // Windows // $cmd = "ls"; // Linux, UNIX & Mac

exec (escapeshellcmd ($cmd), $output, $status);

if ($status) echo "exec Komandası yerinə yetirilməmiş"; else{ echo " <pre>"; foreach ($output as $line) echo htmlspecialchars (" $line\n"); echo "</pre>";}? > htmlspecialchars funksiyası sistem tərəfindən qaytarılan istənilən xüsusi simvolların çevrilməsi məqsədi ilə çağırılır ki , nəticəni qaydaya salaraq başa düşülən və HTML kodu kimi düzgün əks etdirsin. Bu proqramın işə salınması nəticəsində əməliyyat sistemdən asılı olaraq aşağıdakı informasiya (dir Windows-komandasından istifadə vaxtı alınmış) çıxarılacaq:

Volume in drive C is Hard Disk Volume Serial Number is DC63-0E29

Directory of C:\Program Files (x86) \Zend\Apache2\htdocs 09/02/2014 12:03 <DIR>. 09/02/2014 12:03 <DIR>.. 28/04/2013 08:30 5,336 chars.php 12/02/2012 13:08 1,406 favicon.ico 20/01/2014 12:52 4,202 index.html 09/02/2014 11:49 76 info.php 21/03/2013 09:52 110 test.htm 01/04/2013 13:06 182,459 test.php 6 File (s) 193,589 bytes 9 Dir (s) 1,811,290,472,448 bytes free exec funksiyası üç arqument qəbul edir. • Öz komandasını (əvvəlki nümunədə bu $cmd). • sistem komandanın icrası nəticəsində alınan informasiyanı yerləşdiriləcəyi massiv (əvvəlki nümunədə bu $output). • çağırışın qaytarılan statusun saxlanılması üçün dəyişən (əvvəlki nümunədə bu $status). İstəkdən asılı olaraq $output və $status parametrləri ixtisara salmaq olar, amma bu zaman çağırış nəticəsində yaradılmış çıxış informasiyası haqqında heç bir məlumat olmayacaq, hətta funksiyasının icrası barədə xəbərdar olmayacaqsınız. Həmçinin diqqəti escapeshellcmd funksiyasının tətbiqinə yönəldin. Exec funksiyasının çağırışı zamanı bu funksiyadan daim istifadə etmək vərdişini yaratmaq arzuolunandır, çünki, bu funksiya komanda sətrinin tərkibini zərərsizləşdirir, o hadisədə təsadüfi komandaların icrasının (ifasının) qarşısını alaraq, əgər onların daxil etməsinin imkanı istifadəçiyə verilirsə.

Bir qayda olaraq, veb-hostlarda sistem çağırışlarının funksiyaları ümumi istifadə edilən təhlükəni (təhdidi) təhlükəsizlik sisteminə təqdim edənlər kimi qadağan edilmişdir. İmkan daxilində bütün məsələlər PHP vasitələriylə həll etməyə və yalnız sistemə ekstremal ehtiyac vaxtı (yanında) birbaşa müraciət etməyə çalışmaq lazımdır. Bundan başqa, bilməlisiniz ki, sistemə müraciət kifayət qədər yavaş

yerinə yetirilir və əgər proqram işə salınmaya hesablanmışsa, onda həm Windows üçün, həm də Linux/UNIX üçün çağırışın iki növdə reallaşdırılmasını təmin etmək lazımdır.

XHTML və ya HTML5

Bir halda ki, (çünki,) XHTML sənədləri ciddi verilmiş rəsmiləşdirməyə (tərtibata) malik olmalıdır, onlar parsinq standart XML-parserlərdən istifadəylə keçirilə (aparıla) bilər, HTML sənədlərindən fərqli olaraq, hansılar ki, HTML-ın altında daha az vasvası, xüsusi olaraq uyğunlaşdırılmış parser tələb olunur. Buna görə XHTML məşhurluqları fəth etmədi (qazanmadı), və nə vaxt ki, yeni standartı hazırlamaq vaxtı gəldi, W3C üstün tutmanı XHTML2-in yeni standartına vermədi, və (amma) HTML5 dəstəyinə.

HTML5 HTML4 kimi bir çox xüsusiyyətə malikdir, həm də XHTML, amma bu halda o daha çox daha sadə istifadədə və yoxlamaya daha az ciddidir və, bizim həzzimizə, indi HTML5 (müxtəlif ciddiləri, keçidləri daha əvvəl tələb edilənlərin yerinə və tiplərin kadrirovannıx-ı) sənədinin başlığında elan edilən sənədin yalnız bir tipi var məhz: <! Html-ın Sadə sözünün DOCTYPE html>-ı kifayətdir, brauzerə bildirmək üçün ki, sizin veb səhifəniz HTML5 üçün hazırlanmışdır, və, bir halda ki, (çünki,) ən məşhur brauzerlərin bütün ən son versiyaları 2011-ci ildən təxminən başlayaraq HTML5-spesifikasiyaların əksəriyyətini dəstəkləyir, sənədin bu tipi, bir qayda olaraq, yeganə (bircə) lazımdır, əgər, əlbəttə, köhnəlmiş brauzerlərin xidmətinin xeyrinə seçimi etməmək. Bütün hədəflər və niyyətlər üçün, HTML-sənədlərin yazılışı vaxtı (yanında) veb-istehsalçılar köhnə tiplər və XHTML-sənədlərin (məsələn, istifadə
 sadə teqin yerinə
) sintaksisini sakitcə məhəl qoymaya bilərlər. Amma əgər XHTML-a əsaslanan çox köhnə brauzerlərə və ya hər hansı qeyri-adi əlavəyə (proqrama) xidmət etmək lazım olacaqsa onda o haqda informasiyanı, necə bu etmək, http://xhtml.com ünvanında tapmaq olar.

Suallar (məsələlər)

Sual (məsələ) dəyişikliyin 7.1 Hansı spesifikatoru üzən nöqtəylə sayın təsviri üçün printf funksiyada (rolda) istifadə etmək lazımdır? Sual (məsələ) printf-in 7.2 Hansı təlimatı "Happy Birthday" sətirinin qəbulu və "**Happy" sətirinin nəticəsi (çıxardılması) üçün istifadə edilmiş ola bilər? Sual (məsələ) 7.3 Hansı alternativ funksiya printf-dən informasiyanın verilməsi üçün brauzerə istifadə etməmək lazımdır, və (amma) dəyişənə? Vaxt və tarix üçün UNIX vaxtının

qiymətini yaratmaq Kimi sual (məsələ) 7.4, şəklində təqdim edilmişlər (təsəvvür edilmişlər) "7:11am May 2nd, 2016"? Sual (məsələ) fayla girişin 7.5 Hansı rejimi fopen funksiyada (rolda) istifadə etmək lazımdır, onun ölçüsünün kəsilməsiylə və faylın başlanğıcına göstəricinin (siyahının) quraşdırılmasıyla oxumanın və yazının rejimində faylı açmaq üçün? Sual (məsələ) 7.6 Hansı PHP-komanda file.txt faylının silinməsi (uzaqlaşdırması) üçün tətbiq etmək lazımdır?
Sual (məsələ) 7.7 Hansı PHP-funksiya tamamilə bütün faylın oxunması üçün istifadə olunur və hətta çıxartma üçün o Ümumdünya hörümçək torundan? Sual (məsələ) 7.8 hansı superqlobal dəyişən PHP-da serverə yüklənmiş fayllar haqqında məlumatlar olur? Sual (məsələ) 7.9 Hansı PHP-funksiya sistem komandalarını buraxmağa icazə verir? Sual (məsələ) 7.10 HTML5-də teqlərin növbəti stillərindən Hansısı daha üstün: <hr> və ya <hr />? Bu suallara (məsələlərə) cavablar əlavədə (proqramda) A tapmaq olar, "7-ci fəsil məsələlərinə Cavablar" bölməsində.